Računalniški praktikum
2. del

Matjaž Kljun, Branko Kavšek

Študijsko gradivo v elektronski obliki

Univerza na Primorskem, Pedagoška fakulteta Koper

2007

Matjaž Kljun, Branko Kavšek

Računalniški praktikum – 2. del

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004(075.8)(076)

KLJUN, Matjaž
Računalniški praktikum [Elektronski vir] : študijsko gradivo v elektronski obliki / Matjaž Kljun, Branko Kavšek. - Koper : Pedagoška fakulteta, 2007

Način dostopa (URL): http://www.pef.upr.si/gradiva

Dosedanja vsebina:
Del 2. - 2007

ISBN 978-961-6528-65-8 (zv. 2)

1. Kavšek, Branko

231430400

Kazalo

Seminarske vaje

Laboratorijske vaje

Osnove HTML

2006/2007

Matjaž Kljun

Zgodovina

- 1989: Tim Berners-Lee napiše HTML jezik za spletne objave
- March 1993: Lou Montulli objavi brskalnik Lynx različice 2.0a
- 1993: Mosaic
- 1994: Prva World Wide Web konferenca v Ženevi z eno glavnih tem HTML

Zgodovina

- 1994: Internet Engineering Task Force (IETF) organizira HTML delovno skupino
- 1994: objavljena 2. različina HTML specifikacije
- 1994: Netscape
- 1994: The World Wide Web Consortium (W3C)
- 1995: Microsoft IE
- 1997: HTML 3.2 is ready

Značke

- HTML je del družine označevalnih jezikov (več v 2. letniku pri UPO) kot so DGML, XML, XHTL
- Značka označuje lastnosti besedila, ki ga obdaja

  <znacka> besedilo </znacka>

- Značke se piše z malimi črkami (lahko tudi velikime, vendar se taka praksa opušča)

Značke in lastnosti

- Vsaka značka ima nabor lastnosti
- Lastnosti pišemo znotraj značke:

  lastnost="vrednost_lastnosti"

- Vrednost vedno pišemo z dvojnimi narekovajo razen v primerih, ko vrednost sama vsebuje dvojne narekovaje

  lastnost='To je "vrednost" '

HTML dokument - struktura

• Začetek dokumenta	• <html>
Glava	<head>
Naslov strani	<title>Naslov</title>
	<meta podatki>
Konec glave	</head>
Začetek telesa	<body>
Tukaj se nahaja vsebina strani	Besedilo strani
Konec telesa	</body>
Konec HTML dokumenta	</html>

Značke v telesu

• Naslovi	• Naslov 1 .... Naslov 6
Poglavje	<p>Poglavje<p>
Citat	<cite>Citat</cite>
Nova vrstica	 (nima zaključne značke)
Preformatirano besedilo	<pre>Koda</pre>
Črta	<hr>
Komentar	<!-- komentar -->

Končnica dokumenta

- Poznamo dve končnici HTML dokumentov: html in htm.
  Slednji je bil v uporabi zaradi OS, ki so omogočali samo končnice dolge 3 znake.
- Dela oboje.
- Končnica html je bolj opisna
- Dokument, ki ga spletni strezniki najprej prikažejo so: index.html, index.htm, index.asp in index.php

HTML nadaljevanje

2006/2007

Matjaž Kljun

Povezave

- Značka URL povezave:
  Besedilo povezave

  href: samo dokument
  alt: besedilo, ki se izpiše, ko gremo z miško čez povezavo
  target: kako se nova povezava odpre (isto okno, novo okno, ...)

Povezave na določeno besedilo

- Povezave na določen del besedila. Ta mora biti označen:
  Opis //to ni povezava
- Povezava sama zgleda tako:
  Klikni za opis, ki se nahaja na dnu istega dokumenat
  Klikni za opis, ki se nahaja na dnu drugega dokumenat

Povezava na elektronsko pošto

- Značka

  naslov@streznik.si

  S klikom se odpre privzeti program za elektrosnko pošto.

Slike

- Značka slike:

  <img src="slika.gif" alt="To je slika">

- Slike: JPG, GIF, PNG
  JPG predvsem za fotografije, GIF, PNG (transparentnost) za gumbe, okrazske, logotipe.

Relativna in absolutna pot

- Absolutna

  Povezava
  <img
  src="/home/studenti/pef/jaz/public_html/slike/slika.gif">
  <img src="http://www.streznik.si/slikica.gif">

- Relativna (relativno nahajališče glede na dokument ki ga beremo):
  Povezava
  <img src="slike/slika.gif">
  <img src="../public_html/slike/slika.gif">

Tabele

- Značke

```
<table>
   <tr>
      <td>Prva celica prve vrstice</td>
      <td>Druge celica prve vrstice</td>
   </tr>
   <tr>
      <td>Prva celica druge vrstice</td>
      <td> </td>
   </tr>
</table>
```


Tabele z glavo

```
<table>
   <tr>
      <th>Prva vrstica</th>
      <td>Prva celica prve vrstice</td>
      <td>Druge celica prve vrstice</td>
   </tr>
   <tr>
      <th>Druga vrstica</th>
      <td>Prva celica druge vrstice</td>
      <td> </td>
   </tr>
</table>
```


Lastnosti tabele (celice)

- Okvir
  border="1“ bordercolor="red“ background="slika.jpg"
- Podlaga
  color="red“ ali color="#cchh99“
- Razmaki
  cellpadding="3“ (med besedilom in robom)
  cellspacing="3“ (med celicami)
- Poravnava
  align="center“ (left, right)
- Lepljenje:
  colspan="2", rowspan="2"

Gnezdenje tabel

- Tabela v tabeli:

```
<table border="0">
   <tr>
      <td>
        <table> <tr>  <td> Sama celica </td>  </tr> </table>
      </td>
      <td>Druge celica prve vrstice</td>
   </tr>
</table>
```

- Uporabni pri oblikovanju

Predavanje 2 RP 2006/2007 Matjaž Kljun 10

HTML obrazci

2006/2007
Matjaž Kljun

Obrazci

- Namenjeni zbiranju uporabnikovih vnosov

```
<form>
  <input>  --> imamo različne elemente obrazcev
                v katere uporabnik lahko vnese razne informacije
  <input>
</form>
```


Besedilne značke

- Enovrstično besedilo

```
<form>
   First name: <input type="text" name="firstname"> <br>
   Last name: <input type="text" name="lastname">
</form>
```

- Večvrstično besedilo

```
<textarea rows="10" cols="30"> </textarea>
```

Značke izbire

- Radijski gumb (ena sama izbira)

```
<form>
   Spol:
   <input type="radio" name="spol" value="M"> Moški <br>
   <input type="radio" name="spol" value="Z"> Ženski
</form>
```

- Izbirna polja (več izbir)

```
<form> Kako ponavadi potujete
  S kolesom <input type="checkbox" name="vozilo" value="Kolo"><br >
  Z avtom: <input type="checkbox" name="vozilo" value="Avto"><br>
  Z letalom <input type="checkbox" name="vozilo" value="Latalo">
</form>
```


Značke izbire

- Padajoči meni

```
<form> Izberite od kod ste:
  <select name="dezele">
     <option value="pri">Primorska</option>
     <option value="not">Notranjska</option>
     <option value="gor">Gorenjska</option>
     <option value="dol">Dolenjska</option>
     <option value="sta">Štajerska</option>
     <option value="pre">Prekmurje</option>
     .....
  </select>
</form>
```


Gumbi

- Z gumbi potrdimo ali izbrišemo vnesene vrednosti:

```
<form>
   Ime: <input type="text" name="user"><br>
   <input type="submit" value="Potrdi">
   <input type="reset" value="Zbriši">
</form>
```

Akcije

- Vrednosti obrazcev pošljemo v obdelavo „programu“. V našem primeru je to datoteka *obdelaj.php:*

```
<form name="input" action="obdelaj.php" method="get">
  Elektronski naslov: <input type="text" name="enaslov">
  <input type="submit" value="Pošlji">
</form>
```


Metode pošiljanja

- GET - Podatki se pošljejo z URL naslovom (ki je lahko tudi omejen). Uporabljamo ga, ko z njegovo uporabo ne spreminjamo okolja. Primer: iskalniki. Ko vnesemo nekaj v polje in kliknemo „Išči“ s tem ne spremenimo ničesar nikjer.
- POST - Podatki poslani preko zaglavja, količina ni omejena. Uporabljamo, ko poslani podatki spremenijo nekaj: bazo podatkov, pošljejo el. naslov.

GET in POST

- HTTP GET: http://somehost/update?prevName=Raj&preName=Thej
  [http headers]
  [http body (empty)]

  <form action="update" method="get">

- HTTP POST: http://somehost/update
  [http headers]
  prevName=Raj&preName=Thej

  <form action="update" method="post">

GET v/s POST

- 1. POST je varnejši kot GET, ker se spremenljivke ne pokažejo v URL naslovu.
- URL naslovi so omejeni. Na nekazterih strežnikih že z 256 znaki, povečini pa z 4000 znaki in tako ne zmorejo velike količine podatkov v obrazcih.
- Z osvežitvijo spletne strani se GET metoda sama od sebe obnovi (ponovno požene program brez uporabnikovega vedenja).
- Pri post metodi uporabnika pred osvežitvijo brskalnik vpraša, če želimo ponovno pognati program z že vnešenimi podatki.
- GET ne more sprejeti ne-besedilne podatke (npr.: nalaganje slik).

Predavanje 3 RP 2006/2007 Matjaž Kljun 10

CSS

2006/2007

Matjaž Kljun

CSS

- CSS - Cascading Style Sheets ali Prekrivni slogi
- Slog definira izpisovanje/izrisovanje HTML elementov
- Stili so bili dodani HTML 4.0 ra resevanje težav z oblikovanjem (število spletnih strani je lahko zelo veliko)

Zakaj CSS

- Ločitev oblike in vsebine
- Manjši čas nalaganja
- Večja kontrola nad vsebino
- Enakost dokumentov
- Hitrejše urejanje ogromne količine dokumentov

Vključevanje v HTML

- v samih značkah (inline)

```
<p style="color: #000088;">modro besedilo.</p>
```

- vstavljen samostojno v glavi HTML (embedded)

```
<style type="text/css">
<!-- p {color: #000088;} //--></style>
```

- v lastni datoteki (linked): najpogostejša oblika (v glavi HTML)

```
<link rel="stylesheet" href="/path/styles.css"
type="text/css">
```

- uvožen (@import) – podobno kot samostojno v glavi

```
<style type="text/css">
<!-- @import url(pathname/stylesheetname.css);
//--></style>
```


Primer Koktajlov

Spletna stran s CSS dokumentom

Primer koktajlov

Coctail Tequila s soljo in limono

Sestavine

- Tequila
- pol limone
- sol

Priprava

Pri tequili poznamo nekaj različnih načinov pitja. Eden najpopularnejših je prav gotovo tequila, limona in sol. Z limoninim sokom si ovlažimo hrbtišče dlani, posolimo in oblizemo. Čudovit okus popaknemo s tequilo. Za konec se še enkrat osvežimo z limono.

Serviranje

Tequilo serviramo hladno.

- Breskov cocktail
- Kivijev coctail
- Coctail s Camembertom

Slika:

Spletna stran brez oblikovanja

Vprašanje

- Kaj se zgodi, če imamo 100 receptov koktajlov in vsakega v svoji datoteki? Lažje popraviti eno ali 100 datotek?
- Poglejmo si primer oblikovanja strani z recepti, kot izgleda na prejšnjem diapozitivu.

PHP osnove

2006/2007
Matjaž Kljun

PHP in HTML

- PHP je skritpni jezik, ki se izvaja na strežniku; uporabnik dobi le rezultat izvršene kode.
- PHP in HTML lahko med seboj mešamo – to je, PHP kodo vključimo v HTML kodo.
- PHP nam lahko izpisuje HTML.
- Kljub temu, da PHP vključimo v HTML imajo datoteke končnico .php (index.php)

Primer

- PHP kodo ovijemo v <?php /*php koda*/ ?>
- Primer:

```
<html>
    <head><title>Pozdrav</title></head>
    <body>
        <?php echo “<p>Pozdravljeni!</a>“ ?>
    </body>
</html>
```

Poglejte kodo, ki jo dobi brskalnik. To je že sprocesirana koda s strani strežnika. Brskalnik dobi samo HTML kodo.

Osnove

- Vse funkcije najdete na www.php.net
- Spremenljivke ni potrebno definirati in se začnejo vse z znakom $. Ko spremenljivko prvič navedemo jo inicializiramo.

  $stevilo=3;

- *echo* funkcija izpisuje v okno brskalnika.

Spremenljivke

- Logična vrednost: $foo = True;
- Celo število: $a = 1234; $a = -123; $a = 0x1A;
- Relano št: $a = 1.234; $b = 1.2e3;
- Niz: $foo = 'Foo';
- Polje: $arr = array("foo" => "bar", 12 => true);
  echo $arr["foo"]; // izpise bar
  echo $arr[12]; // izpise 1

Spemenljivke

- Razredi

```
class foo {
   function do_foo()   {
       echo "Doing foo.";
   }
}

$bar = new foo; //kreiranje nove instance razreda
$bar->do_foo(); //klicanje funkcije razreda
```

- echo gettype($sprmenljivka); nam izpiše tip

Narekovaji

- Enojni narekovaji:

  echo 'Pozdravljeni '.$ime;
  (echo 'Pozdravljeni $ime';)

  niz se ne procesira – izpiše se tak kot je.
- Dvojni narekovaji:

  echo “Pozdravljeni $ime“;

  niz se procesira – izpišejo se spremenljivke.

Kontrola toka

- if else elseif
- while do-while
- for foreach
- break continue switch
- declare
- return
- require() include()

PHP

- Zgoraj predstavljene osnove bomo nadgrajevali na vajah s primeri.

  Več o PHP najdete na
  http://www.php.net/manual/en/

Osnove mySQL

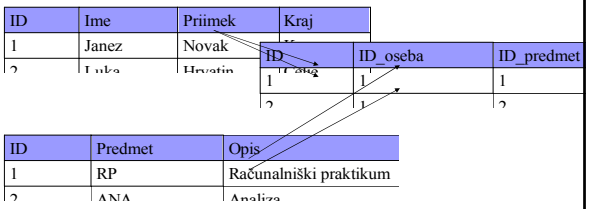
2006/2007
Matjaž Kljun

Podatkovna baza

- je množica med seboj povezanih podatkov – relacijska PB – relacije med podatki,
- podatki organizirani v tabele,
- stolpci opisujejo vsebino,
- vrstice vsebujejo posamezne zapise,
- vsak zapis je enolično določen,
- uporaba SQL (structured query language).

Primer

- Imamo tri tabele, ki so med seboj povezane preko ključev – ključ enolično določa zapis v tabeli



Primer dnevnika

- Dve tabeli:
  v prvi bomo hranili zapise v dnevnik,
  v drugi pa komentarje na zapise.

id_blog	ime_avtor	priimek_avtor	e-naslov	nasl
1	Janez	Novak	jn@me.si	Kaj
2	Peter	Pavel	pp@me.je	Kam

id_komntar	id_blog	ime
1	1	Jože
2	1	Janez

Imamo dva komentarja na prvi zapis v dnevnik. Vsak zapis v dnevnik ima lahko poljubno mnogo komentarjev. Kateri komentar spada h kateremu zapisu vemo s pomočjo ključa (id_blog, id_komentar).

Dostopanje do mySQL

- Potrebujemo: ime strežnika, ime baze, uporabniško ime in geslo.
- Ukazna vrstica:

  $ mysql -h streznik -p uporabnisko.ime -p geslo

  se poveže na lokalnem strežniku kot uporabnik uporabnisko.ime
- Uporabniški vmesniki

  http://www.studenti.pef.upr.si/phpmyadmin

Interni mySQL ukazi

- SHOW DATABASES
  Prikaz baz podatkov na strežniku MySQL
- SHOW TABLES FROM db:
  Prikaz tabel v bazi podatkov, v kateri se trenutno nahajamo
- SHOW COLUMNS FROM tabela
  Prikaz vseh stolpcev (polj) v tabeli
- DESCRIBE tabela
  Opis vseh polj iz tabele
- SHOW GRANTS FOR uporabnik
  Prikaz pravic uporabnika
- SELECT DATABASE(db)
  Zamenjava baze podatkov

SQL ukazi – baza podatkov

- CREATE DATABASE ime_bp;
  Ustvarimo bazo podatkov z imenom ime_bp.
- CONNECT ime_bp;
  Povežemo se z bazo podatkov ime_bp – s tem jo lahko začnemo urejat.
- DROP DATABASE ime_db
  Ukaz pobriše celotno bazo podatkov ime_bp.
- GRANT <pravica> ON <objekt> TO <uporabnik>
  Ukaz dodeli pravico *pravica* za *objekt* uporabniku *uporabnik*.

  Pravice so lahko:
  ALL PRIVILEGES: uporabnik ima vse pravice nad bazo podatkov·
  USAGE: uporabnik nima nobene pravice
  ALTER, CREATE, DELETE, DROP, GRANT, INDEX, INSERT, DELETE, SELECT: pravica za izvajanje posameznega ukaza.

SQL ukazi – izdelava tabele

- CREATE TABLE tab (ime_1stolpca lastnosti,
  ime_2stolpca lastnosti,
  ...
  ime_Nstolpca lastnosti);
- Ukaz ustvari tabelo z imenom tab, ki vsebuje N stolpcev tako kot smo jih definirali.
- Osnovne lastnosti stolpcev (tj. opisi prilastkov) so lahko:
  - tip podatka,
  - obveznost podatka,
  - privzeta vrednost podatka,
  - informacija ali gre za ključ.
- Tip podatka je potrebno obvezno določiti za vsak stolpec posebej. Osnovni tipi podatkov:
  INTEGER: celo število od –231 do 231,
  FLOAT(N,M): število z največ N znaki pred in največ M znaki po decim. vejici,
  VARCHAR(N): besedilo največ dolžine N (N znakov)
  CHAR(N): besedilo natančno dolžine N (N znakov)
  TEXT: Polje z največ 216–1 znaki,
  MEDIUMTEXT: Polje z največ 224– 1 znaki,
  DATE: datum,
  TIMESTAMP: čas (z vključenim datumom).
- Obveznost podatka:

  NULL: vrednost v polju je lahko tudi prazna,

  NOT NULL: podatek v polju je obvezen.
- Privzeta vrednost podatka:

  AUTO_INCREMENT: lahko uporabimo samo v primeru stolpcev, kjer je tip podatka INTEGER (vrednost v polju se sama povečuje z vnosom novega zapisa v tabelo),

  DEFAULT: določimo privzeto vrednost polja (ob vnosu zapisa v tabelo se polje s to lastnostjo samodejno zapolni s privzeto vrednostjo).
- Informacija ali gre za ključ:

  PRIMARY KEY: polje je tudi glavni ključ tabele (polje, ki je primarni ključ ne sme biti nikoli prazno, torej vedno zahtevamo še lastnost NOT NULL, in mora biti enolično),

  INDEX: index lahko vključuje več polj hkrati in omogoča ustvarjanje »indeksa« nad podatki v tabeli po izbranih poljih, tako da je iskanje podatkov po teh poljih bistveno hitrejše.

SQL – vnos podatkov

- Ukaz vpiše nov zapis v tabelo in sicer v tista polja, ki smo jih navedli. Ukaz seveda ne uspe, če v polja z lastnostjo NOT NULL nismo vnesli nobene vrednosti.

  INSERT INTO tabela
  (ime_1stolpca, ime_2stolpca, ...)
  VALUES (vrednost1, vrednost2, ...);

SQL - poizvedovanje

- Ukaz izbere iz tabele *tabela* samo stolpce iz *seznam_polj* za tiste zapise, ki ustrezajo pogoju *pogoj*.

```
SELECT seznam_polj
FROM tabela
WHERE pogoj;
```

```
SELECT * FROM blog;
SELECT * FROM blog WHERE datum > 2007-01-01;
```

Predavanje 6 RP 2006/2007 Matjaž Kljun 10

SQL - brisanje

- Brisanje podatkov v tabelah
  Ukaz pobriše vse zapise v tabeli, ki ustrezajo določenemu pogoju.

```
DELETE FROM tabela
WHERE pogoj
```

Predavanje 6 RP 2006/2007 Matjaž Kljun 11

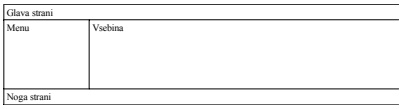
Racionalizacija spletne strani

2006/2007

Matjaž Kljun

Struktura strani

- Poglejmo kako smo si zamislili strukturo strani na vajah



Zakaj racionalizirati

- Če želimo naprimer besedo, ki se nahaja na vseh straneh spremenit, moramo popravljati vse strani.

  Naj bo to *Noga strani*. Če jo hranimo v svoji datoteki, ki jo vključimo na vseh straneh, jo popravljamo samo na enem mestu.

Kaj racionalizirati

- Stvari ki se ponavljajo:

  - HMTL glava (<head>)
  - Glava dokumenta (logotip)
  - Menu
  - Noga dokumenta ((c))

PHP

- include(“glava.php“);

  S to funkcijo vključimo drugi dokument v našega.

  V svojih datotekah lahko hranimo tudi funkcije, ki jih pogosto rabimo, splošne spremenljivke, itd.

Kako ločiti našo stran z vaj

- glava.php (HTML zaglavje)
- logo.php (glava dokumenta oz. logotip)
- meni.php (meni spletne strani)
- noga.php (noga spletne strani)

```
<?php
include('glava');
include('logo.php');
include('meni');
?>
VSEBINA
<?php
include('noga');
?>
```

Drugi primer

- Lahko pa cel okvir ki se ne spreminja hranimo v eni datoteki in samo v vsebinski del kličemo dele, ki se spreminjajo:

```
<!-- html do vsebine -->
<?php
if ($stran==“dodaj“) include('dodaj.php');
if ($stran==“brisi“) include('brisi.php');
if ($stran==“prikazi“) include('prikazi.php');
if ($stran==““) include('prvastran.php');
?>
<!-- html do konca -->
```

Zaključek spletne strani

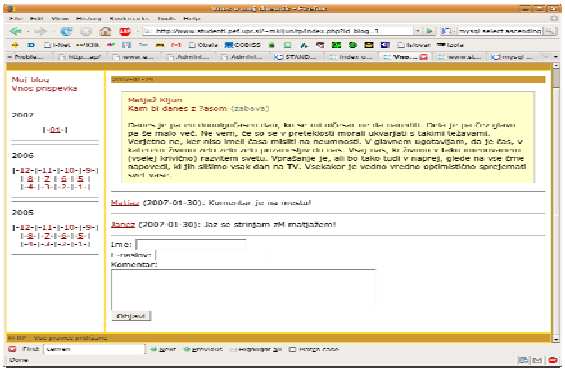
2006/2007
Matjaž Kljun

Kaj še manjka

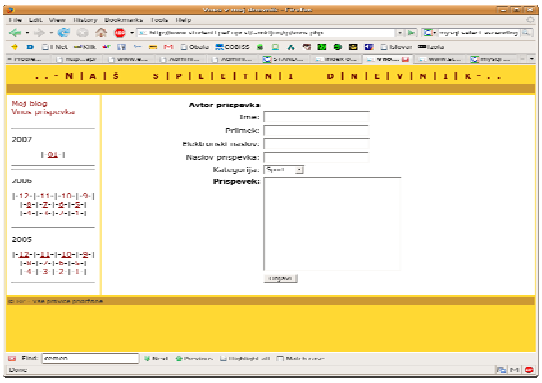
- Vnos komentarja
- Menu (izpis po mesecih)
- Izpis komentarjev pod vnosom v dnevnik
- Dopolni glavo

Blog še zdaleč ni končan. Za vnose v dnevnik bi lahko uporabili avtentikacijo uporabnika, pri komentarjih bi lahko še preverjali pravilnost vnosov, pri vnosih bi lahko dodali sliko za poživitev, blog bi lahko bil večjezičen, ... Ravno tako ni koda optimizirana, ker je prilagojena bolj učenju. Podatki o bazi bi lahko bili v svoji datoteki (če premaknemo vse na drugi strežnik ni potrebno spreminjat vseh datotek), ...

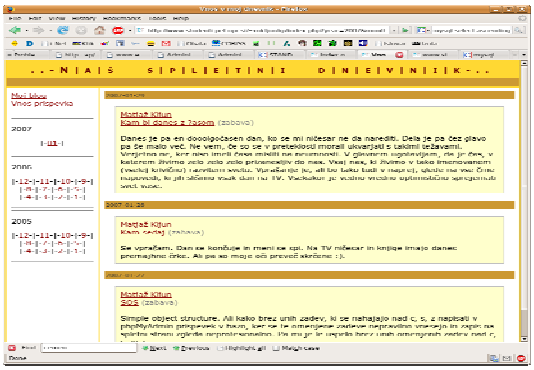
Zadnji vnos



Vnos



Vnosi enega mesca



Regularni izrazi

2006/2007
Matjaž Kljun

Kaj so RI

- Pri blogu smo pri vnašanju v blog preverjali, če so vnosi v polja zahtevane oblike:

```
ereg("([A-Z]{1})[[:alpha:]]", $ime_avtorja);

ereg("^[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+"."@".
     "[-!#$%&\'*+\\/0-9=?A-Z^_`a-z{|}~]+\.".
"[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+$",
     $elektronski_naslov);
```

Pri prvem preverjamo, če ima spremenljivka $ime_avtorja veliko začetnico in ostale črke iz abecede.

Pri drugem primeru preverjamo sintaktično pravilnost vnešenega elektronskega naslova.

Osnovna sintaksa

- Z regularnimi izrazi opisujemo nize znakov.

  (a|b)*ccc

  opisuje nize, ki se začnejo s poljubnim številom `a`-jev in `b`-jev in se končajo s tremi `c`-ji.
- znak `|` loči alternative
  `a|b` je "`a` ali `b`";
- znak `*` pomeni nič ali več ponovitev izraza pred znakom
  `(a|b)*` = "poljubno število ponovitev: `a` ali `b`."
- (`r`) z oklepaji omejimo podniz
- na koncu še trije `c`-ji
- Primeri:

```
ccc                 cccccc (ne)
abababaccc          cccaababa (ne)
bbbbaccc            accc
```

Primeri

- Primeri regularnih izrazov in nizi, ki jih opisujejo:

  (F|f)rank (S|s)inatra

```
Frank Sinatra
Frank sinatra
frank Sinatra
frank sinatra
```


Primeri

- (Frank Sinatra)|(frank sinatra)

```
Frank Sinatra
frank sinatra
```

- 0|((1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*)
  Niz `0` ter vsi nizi števk, ki se ne začnejo z ničlo.
- (a*)b(a*)b(a*)b(a*)
  Vsi nizi `a`-jev in `b`-jev, v katerih se b pojavi natanko trikrat.

Sintaksa

- Pogosto potrebujemo regularni izraz, ki opisuje vse črke abecede. To je seveda izraz

  a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

  ki pa je predolg in nepraktičen.

```
[[:alpha:]](vse črka) [a-z](vse male črke a do z)
```

- Kadar želimo opisati nekaj znakov, lahko uporabimo krajšo notacijo:

  [`abc`] Znak `a`, `b` ali `c`.
- [^`abc`] Katerikoli znak *razen* znakov `a`, `b` in `c`.

Primeri

- 0|([1-9][0-9]*)
  Niz `0` ter vsi nizi števk, ki se ne začnejo z ničlo.
- [A-Z][a-z]*
  Besede, ki se začnejo z veliko začetnico.
- [A-Da-z]*
  Nizi, ki sestojijo iz črk `A`, `B`, `C` in `D` in malih črk, na primer `aaaBDBlksadzzojC`.

Sintaksa

- \d vsa števila; [0-9].
- \D vse neštevila; [^0-9].
- \s vse nevidne (prazne) znake; [\t\n\r\f\v].
- \S vse vidne znake; [^ \t\n\r\f\v].
- \w vse črke in številke; [a-zA-Z0-9_].
- \W vse razen črk in številk; [^a-zA-Z0-9_].
- \| znak |; [|]

Sintaksa

- **.** Matches any single character. Into [ ] this character has its habitual meaning. For example, "a.cd" matches "abcd", "a..d" matches "abcd".
- **[ ]** Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] matches any lowercase letter. These can be mixed: [abcq-z] matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does [a-cq-z].

  The '-' character should be literal only if it is the last or the first character within the brackets: [abc-] or [-abc]. To match an '[' or ']' character, the easiest way is to make sure the closing bracket is first in the enclosing square brackets: [][ab] matches ']', '[', 'a' or 'b'.
- **[^]** Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter. As above, these can be mixed.
- **^** Matches the start of the line (or any line, when applied in multiline mode)

Sintaksa

- $ Matches the end of the line (or any line, when applied in multiline mode)
- ( ) Defines a "marked subexpression". What the enclosed expression matched can be recalled later. See the next entry, \n. A "marked subexpression" is also a "block". This feature is not found in some instances of regex. In most Unix utilities (such as sed and vi) a backslash must precede the open and close parentheses.
- \n Where n is a digit from 1 to 9; matches what the nth marked subexpression matched. This construct is theoretically irregular and has not been adopted in the extended regular expression syntax.
- * A single character expression followed by "*" matches zero or more copies of the expression. For example, "ab*c" matches "ac", "abc", "abbbc" etc. "[xyz]*" matches "", "x", "y", "zx", "zyx", and so on.

  * \n*, where n is a digit from 1 to 9, matches zero or more iterations of what the nth marked subexpression matched. For example, "(a.)c\1*" matches "abcab" and "abcabab" but not "abcac".

  * An expression enclosed in "\(" and "\)" followed by "*" is deemed to be invalid. In some cases (e.g. /usr/bin/xpg4/grep of SunOS 5.8), it matches zero or more iterations of the string that the enclosed expression matches. In other cases (e.g. /usr/bin/grep of SunOS 5.8), it matches what the enclosed expression matches, followed by a literal "*".

Sintaksa

- + A single character expression followed by "+" matches one or more copies of the expression. For example, "ab+c" matches "abc", "abbbc" etc. "[xyz]+" matches "x", "y", "zx", "zyx", and so on.

  * \n+, where n is a digit from 1 to 9, matches one or more iterations of what the nth marked subexpression matched.

  * An expression enclosed in "\(" and "\)" followed by "+" is deemed to be invalid.

- {x,y} Match the last "block" at least x and not more than y times. For example, "a\{3,5\}" matches "aaa", "aaaa" or "aaaaa". Note that this is not found in some instances of regex.

Okrajšave

- [:upper:] [A-Z] velike črke
- [:lower:] [a-z] male črke
- [:alpha:] [A-Za-z] velike in male črke
- [:alnum:] [A-Za-z0-9] velike, male črke in števila
- [:digit:] [0-9] števila
- [:xdigit:] [0-9A-Fa-f] šestnajstiška števila
- [:punct:] [.,!?:...] ločila
- [:blank:] [\t] tabulator
- [:space:] [\t\n\r\f\v] vsi „nevidni“ znaki
- [:cntrl:] znak control
- [:graph:] [^ \t\n\r\f\v] vsi vidni znaki
- [:print:] [^\t\n\r\f\v] vsi vidni znaki in presledek

SED

2006/2007
Matjaž Kljun

SED

- `sed` (stream editor) o
- omogoča spreminjanje delov besedila v datotekah iz ukazne vrstice.
- ni interaktiven: bere vrstico za vrstico iz standardnega vhoda, na vsaki izvrši dano operacijo in jih izpisuje na standardni izhod.
- hiter: ker bere, spreminja in izpisuje vrstico za vrstico, ni potrebno prebrati celo datoteko v pomnilnik

Kaj omogoča

- lahko spreminjamo in zamenjujemo besede,
- brišemo prazne vrstice,
- izpisujemo izbrane vrstice
- ...

- pogosta uporaba v cevovodu

Primer:

Sintaksa

- Oblika ukaza

  sed [-n] [-e script] [-f sfilename] [filename ...]

  Opcija `-n` ne izpisuje na standardni izhod.
  Opcija `-f` bere ukaze za urejanje iz datoteke `sfilename`.

  Zapis `script` vsebuje urejevalne ukaze naslednje oblike:

  [naslov[,naslov]]funkcija[argumenti]

Predavanje 10 RP 2006/2007 Matjaž Kljun 4

Primeri

- sed -e 's/slika/miza/' besedilo.txt

  Vsako pojavitev niza "`slika`" v datoteki `besedilo.txt` zamenja z nizom "`miza`" in izpiše rezultat (vrstica za vrstico).
- sed -e '/^$/d' dat1 > dat2

  Ukaz zbriše vse prazne vrstice (operator "`d`") iz datoteke `dat1` in rezultat shrani v datoteko `dat2`.
- sed -e 's/janez/Janez/g' < dat1

  Ukaz bere besedilo iz datoteke `dat1`, zamenja vse pojavitve niza "`janez`" z "`Janez`" (če ne bi uporabili operatorja "`g`", bi se zamenjala samo prva pojavitev). Izhodno besedilo se izpiše na standardni izhod.

Predavanje 10 RP 2006/2007 Matjaž Kljun 5

AWK

2006/2007
Matjaž Kljun

AWK

- Programski jezik namenjen obdelavi besedil
- 1977 napisan v enem tednu
- Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan;
- Pisan v velikimi tiskanimi črkami
- Popularen zaradi zgodnjega nastanka in možnosti uporabe v cevovodih

Sintaksa

- Enovrstični ukaz

  awk <search pattern> {<program actions>}

  AWK gre skozi vrstice datoteke in pri tistih, ki ustrezajo iskalnemu nizu izvede dejanje (akcijo).

- AKW program

```
awk 'BEGIN          {<initializations>}
        <search pattern 1> {<program actions>}
        <search pattern 2> {<program actions>}
     END            {<final actions>}'
```

  BEGIN vrstica nam pred procesiranjem datoteke izvede inicializacijo. END pa po branju datoteke izvede še dodatne akcije (npr. dodatni izpisi).

Potek izvedbe

1. Naloži skripto in jo prevede v interno obliko, to pa hitro izvaja,
2. Izvede stavke v bloku BEGIN, če ta obstaja,
3. Odpre vhodno datoteko,
4. Prebere vrstico vhodne datoteke,
5. Izvede vsa dejanja, za katera je pogoj resničen,
6. Dokler nismo na koncu vhodne datoteke, se vrne na točko 4,
7. Zapre vhodno datoteko,
8. Izvede stavke v bloku END, če ta obstaja.

Izvajanje

- Če manjka pogoj, se dejanje izvede za vse vrstice.
- Če smo zapisali pogoj, vendar manjka dejanje, awk privzame, da je to izpis tekoče vrstice z ukazom { print; }.
- Ukaz print uporabi privzeti argument, če ga izrecno ne navedemo.
- Program awk bere vhodno datoteko po vrsticah.

  To lahko spremenimo in zahtevamo, da bere zapise, ki so med seboj ločeni z vrednostjo spremenljivke RS. Awk zna še več - zapis je sestavljen iz polj, privzeti ločnik med polji so presledki, privzeti ločnik med zapisi pa nova vrstica. Tako zelo preprosto dosežemo stolpce v datoteki, v kateri so ti med seboj ločeni s presledki.

Primer

- Če nas zanima velikost datotek in podimenikov v dnevniškem imeniku, bomo v ukazni lupini izvedli:

  ls –l /var/log | awk '{print $5, $9;}'

  Dobljeni izpis bo:

```
288913 boot.log
24664 cron
...
3918 xdm-error.log
```

  Z ***$5*** smo se sklicevali na peti stolpec izhoda programa ls s stikalom –l. V petem stolpcu so dolžine datotek v bajtih, v devetem pa imena datotek. Neoblikovani izpis obojega izpiše v vsaki vrstici dolžino datoteke in njeno ime.

  Poglej razliko z navadnim ls ukazom!

Spremenljivke

- Se ustvarijo dinamično ob prvem nastopu v skripti.
- Osnovna podatkovna tipa spremenljivk, znakovni in numerični.
- Pri slednjem je spremenljivka predstavljena kot realno število.
- Privzeta vrednost spremenljivke je prazen niz oziroma nič (0,0), odvisno od mesta, na katerem to spremenljivko prvič uporabimo.

POSEBNE SPREMENLJIVKE

- $X, ki označuje trenutno vrednost Xtega polja v zapisu (Celotno vrstico dobimo z $0)
- Spremenljivka ***NF*** hrani število polj v trenutnem zapisu
- ***Spremenljivka NR*** hrani zaporedno številko trenutnega zapisa (to je enako zaporedni številki vrstice, če nismo spremenili ločnika zapisov)
- Naslovljivo polje: ***Telefon["Gasilci"]=112;***

Kontrola toka

- Krmilne strukture, ki jih awk pozna, so podobne kot v jeziku C. Zgled, ki izpiše število pojavitev vsake besede v vhodni datoteki:

```
{ for ( i = 1; i <= NF; i++){ Besede[i]++; } }
END { for (s in Besede) print Besede[s], s; }
```

Dejanje v prvi vrstici: za vsako vhodno besedo poveča števec pojavitev te besede v tabeli vseh besed.
Druga vrstica: za vsako od besed v tabeli vseh besed program izpiše, kolikokrat se je pojavila.

Primeri

- Skripta, ki bo izpiše tekstovno vhodno datoteko:
{ print $0 } - enako bi storili lahko z less, more, cat

- Nadomestek za program wc (word count), ki izpiše število vrstic, besed in znakov v vhodni datoteki:

```
{ znaki += length($0)
  besede += NF }
END { print NR, besede, znaki }
```

- Vrednost v prvem stolpcu vhodne datoteke in njihovo povprečje izračunamo:

```
{ vsota += $1 }
 END { print vsota, vsota/NR }
```

Primeri

- Vrstice, v katerih se prvo polje razlikuje od prvega polja prejšnje vrstice, izpišemo:

```
$1  != prejšnja { print: prejšnja = $1 }
```

- Vse vrstice med vrsticama, v katerih sta besedi začetek in konec, izpišemo z:

```
/začetek/, /konec/{ print }
```

Enako lahko stori tudi sed.

Na vajah bomo pogledali, več primerov.

Predavanje 11 RP 2006/2007 Matjaž Kljun 10

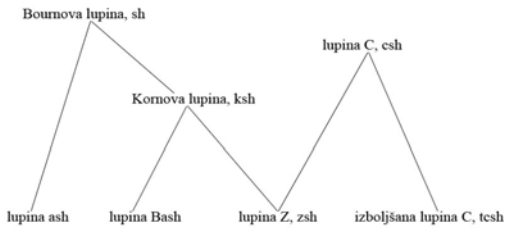
Programiranje v lupini

2006/2007
Matjaž Kljun

Lupine

- Poznamo več lupin. Najpogostejše so
  sh
  csh
  tcsh
  bash
- Vsaka ima svoje značilnosti, vendar je programiranje v vseh v osnovi podobno.

Razvoj lupin



Razčlenjevanje ukaza

- Lupina najprej razreši posebne znake, z nadomestnimi znaki vred. Posebni znak ? bo nadomestil katerikoli ujemajoči znak.
- Lupina osveži zgodovino vnesenih ukazov v pomnilniku in na disku v datoteki, kjer se shranjujejo pretekli ukazi.
- Razreši narekovaje. Besedilo znotraj enojnih narekovajev ne doživi spremembe, spremenljivke med dvojnimi narekovaji pa se razvežejo v svoje pomene.
- Razveže nadomestna imena v osnovna. Nadomestna imena, ki smo jih določili z ukazom »alias«, zamenja z njihovim pomenom.
- Poišče morebitne rezervirane besede. Rezervirane besede vodijo tok programa. Na primer if ali while.
- Izvede uporabljene funkcije.
- Ukazno vrstico razreže v besede in prepozna preusmeritve.
- upošteva podane preusmeritve, na primer > ali >&, jih upošteva in datotekam priredi prave tokove.
- Razporedi posle. Če smo na primer ukaz zaključili z znakom & za izvajanje v ozadju, tak proces ponikne v ozadje.
- Poskrbi za sistemske signale.
- Poišče izvedljive datoteke z imenovanimi programi. Če datotek ne najde, javi napako.
- Imenovane programe izvede.

Skripte

- to so namenski programe, ukazne skripte, s katerimi poenostavimo številne postopke in avtomatiziramo rutinska opravila.
- Vse ukaze, ki jih uporabljamo v ukaznih lupinah, lahko brez sprememb uporabljamo v skriptah .

Bournova lupina

- Leta 1974 jo je napisalo kakih trideset ljudi v Bell Labs, njen poglavitni avtor pa je Stephen R. Bourne.
- Prva poimennovana po avtorjih.
- Privzeta lupina skrbnika sistema Unixa (v linuxu je to ponavadi bash).
- Zaradi njene učinkovitosti in razširjenosti je večina sistemskih skriptov napisana v tem jeziku (apropos in whatis).
- Bournova ukazna lupina ima vgrajen še za današnje razmere skladenjsko močan jezik.
- Poglavitne lastnosti:
  - vgrajena zgodovina ukazov, nadomestna imena, vgrajena aritmetika, dopolnjevanje imen in nadzor poslov.
- Skoraj v celoti navzgor združljiva s Kornovo lupino (večina skriptov teče brez spremembe).
- Največja šibkost pa je skoraj popolno pomanjkanje sredstev, ki olajšujejo interaktivno delo, predvsem kar zadeva priklic nasih prejsnjih ukazov.

Lupina C

- Razvili na Kalifornijski univerzi v Berkeleyju konec sedemdesetih let (Bill Joy). Izdana kot del Unixa 2BSD.
- Osnova njene skladnje jezik C.
- Precej zmogliva a za tedanje osebne računalnike preveč potratna z viri.
- Uporabniki imajo lupino C rajši kot Bournovo, ker lahko uporabljajo številne skladenjske elemente tako kakor v jeziku C.
- Sodobnejša, izboljšana in z lupino C popolnoma združljiva ukazna lupina je tcsh (dodajana ukazna zgodovina in dopolnjevanjem imen datotek).
- Križanec med lupinama Bash in C je lupina zsh. Njena najmočnejša prednost je samodejna pomoč pri izbiri parametrov programov.

Kornova lupina

- Leta 1986 napisal David Korn z AT&T.
- Leta 1988 je postala del uradne distribucije Unixa SVR4.
- Danes teče v sistemih Unix, OS/2, VMS in DOS.
- Komercialna licenca.
- Odprtokodna je okrnjena pdksh in jo najdemo tudi v distribucijah Linuxa.
- Kornova lupine je nadgradnja Bournove.
- Združuje lepe interakcijske prijeme lupine C z močno in bogato skladnjo Bournove lupine.
- Vnesene ukase lupina pomni, tako da jih lahko prikličemo ter zlahka urejamo in spreminjamo.
- Lupina pozna regularne izraze, nadomestna imena, funkcije, nadomestne znake, soprocese, vgrajeno aritmetiko, nadzor poslov in pripomočke za razhroščevanje.

Lupina Bash

- Nastala kot del projekta GNU, sin stem prostodostopna.
- Pobudnik FSF, da je nujna prosto dostopna lupina, ki bo združljiva s standardom POSIX.
- POSIX se je takrat ravnal po zgledu Kornove lupine in jo tako rekoč uzakonil.
- Bash, po angleško Bourne again shell, ponovno Bournova lupina, kar v angleščini zveni kot vnovič porojena lupina.
- Prvi avtor je bil Brian Fox iz Free Software Foundation, trenutno pa njen poglavitni razvijalec in vzdrževalec Chet Ramey z univerze CaseWestern Reserve.
- Privzeta lupina v Linuxu.

Vaja 01: HTML

Primer: osnovni dokument

```
<html>
  <head>
     <title>Naslov</title>
  </head>
  <body>
     Besedilo strani
  </body>
</html>
```

Primer: Lastnosti besedila

```
<html>
  <head>
     <title>Naslov</title>
  </head>
  <body>
     <h1>Naslov 1</h1>
     <h2>Naslov 2</h2>
     <h3>Naslov 3</h3>
     <h4>Naslov 4</h4>
     <h5>Naslov 5</h5>
     <h6>Naslov 6</h6>
     <p>Poglavje s <b>krepkim</b>, <i>ležečim</i> in <u>podčrtanim</u>
        besedilom.
            Kljub novi vrstici v kodi se to
         besedilo ne bo prikazalo v novi vrstici.
         Od tukaj dalje pa ja <br>
           - nova vrstica.
     </p>
     <hr>
     <cite>To je citat</citat>
     <hr>
     <pre>
       #include <stdio.h>
       int main () {
         printf(„Pozdravljen svet!“);
         return 0;
       }
     </pre>
     <!-- komentarje brskalnik ne prikaže -->
  </body>
</html>
```

Najpogostejše značke:

`Značka	Opis
<b>	Krepko besedilo
<big>	Povečano besedilo
	Povdarjeno besedilo
<i>	Ležeče besedilo

<small>	Pomanjšano besedilo
	Povdarjeno besedilo
<sub>	Podpisano besedilo
<sup>	Nadpisano besedilo
<ins>	Vnešeno besedilo
<del>	Zbrisano besedilo
<code>	Računalniška koda
<kbd>	Besedilo s tipkovneice
<samp>	Primer računalniške kode
<tt>	Besedilo tipkarskega stroja
<var>	Spremenljivka
<pre>	Preformatirano besedilo
<abbr>	Okrajšava
<acronym>	Akronim
<address>	Naslov
<bdo>	Smer besedila
<blockquote>	Daljši navedek (citat)
<q>	Krajši navedek (citat)
<cite>	Citat
<dfn>	Definicija

Primer: celotno zaglavje

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
     <title>Naslov</title>
     <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <meta name="Keywords" content="html,w3c,internet">
     <meta name="Description" content="Opis, ki ga prikažejo iskalnik.">
     <meta http-equiv="pragma" content="no-cache">
     <meta http-equiv="cache-control" content="no-cache">
  </head>
  <body>
    Besedilo dokumenta
  </body>
</html>
```

Vaja 2: Nadaljevanje HTML

Primer: sidra

```
<html>
  <head>
     <title>Naslov</title>
  </head>
  <body>
     <p>
       Če bi radi izvedeli kaj več,
       <a href="#opis">kliknite sem za opis, ki se nahaja v
         drugem odstavku</a>.
     <p>

     <p>
      <a href="#opis">Opis</a>
      Opis nečesa, na kar se sklicujemo zgoraj. Na tak
      način lahko delamo kazala knjig.
     </p>
  </body>
</html>
```

Primer: URL

```
<html>
  <head>
     <title>Naslov</title>
  </head>
  <body>
     <p>
       Tukaj je povezava na spletno stran
       <a href="http://www.upr.si">UP</a>.
     <p>
     <p>
       Tukaj pa povezava na spletno stran
       <a href="dokument.html">na istem strežniku</a>
       v istem direktoriju.
     </p>
  </body>
</html>
```

Primer: bolj zakomplicirana tabele

```
<html>
  <head>
     <title>Naslov</title>
  </head>
  <body>

  <table width="60%" bgcolor="#000000" align="center"
         border="0" cellspacing="0">
   <tr><td>

   <table width="100%" cellspacing="1" cellpadding="3">
     <tr bgcolor="#cccccc"> <th> </th>
```

```
            <th>Ponedeljek</th>
            <th>Torek</th>
      </tr>
      <tr bgcolor="#ffffff">
            <th bgcolor="#cccccc">9:00-10:00</th>
            <td>RP</td>
            <td rowspan="2" align="center">Analiza 1</td>
      </tr>
      <tr bgcolor="#ffffff">
            <th bgcolor="#cccccc">10:00-11:00</th>
            <td>Diskretne strukture</td>
            <!-- te celice ni ker je zgornja čez dve <td>Analiza 1</td> -->
      </tr>
   </table>

   </td></tr>
  </table>

</body>
</html>
```

Primer: slike

```
<html>
  <head>
     <title>Naslov</title>
  </head>
  <body>
    <p>To je slika z levo poravnanim besedilom
      <img scr="slika.gif" align ="left">.
       Možna je še desna (right) poravnava. Lahko dodamo še
       širino (width="300"), višino (height="300"), besedilo
       (alt="Opisno besedilo").<p>
     <p>To je slika s sredinsko poravnanim besedilom na vertikali
       <img scr="slika.gif" align ="middle">. Možnosti so še
       zgornja (top) in spodnja privzeta (bottom) poravnava.<p>
     <table border="1">
       <tr>
         <td background="slika.gif"> Celica tabele ima lahko za podlago
           sliko. Tudi cel HTML dokument ima lahko sliko za podlago
           (dodamo lastnost <i>background</i> v značko <i>body</i>).
         <td>
       </tr>
     </table>
  </body>
</html>
```

Vaja 3: Nadaljevanje HTML

Primer: uporaben obrazec za pošiljanje elektronske pošte

```
<html>
 <body>
   <form action="MAILTO:matjaz@pef.upr.si" method="post" enctype="text/plain">
     Pošljite elektronsko pošto na Matjažev e-naslov <br>
     Vaše ime: <input type="text" name="ime" value="" size="20"><br>
     Vaš elektronski naslov:
       <input type="text" name="email" value="" size="40"><br>
     Vsebina sporočila:<br>
      <textarea rows="10" cols="30"> </textarea><br>
     <input type="submit" value="Pošlji">
     <input type="reset" value="Zbriši">
   </form>
 </body>
</html>
```

Primer: vnos dnevniškega zapisa (zasnova BLOG sistema)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Vnos v moj dnevnik</title>
</head>
<body>
  <form method="post" action="obdelaj.php" name="prispevek">
    <table width="100%" border="1" cellpadding="2" cellspacing="2">
      <tr>
        <td align="right">Avtor prispevka</td><td> </td>
      </tr>
      <tr>
        <td align="right">Ime: </td><td><input name="ime"></td>
      </tr>
      <tr>
        <td align="right">Priimek: </td><td><input name="priimek"></td>
      </tr>
      <tr>
        <td align="right">Elektronski naslov: </td>
        <td><input name="email"></td>
      </tr>
      <tr>
        <td valign="top" align="right">Prispevek: </td>
        <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
      </tr>
      <tr>
        <td align="right">Kategorija: </td>
        <td>
          <select name="kategorija">
            <option value="sport">&Scaron;port</option>
            <option value="politika">Politika</option>
            <option value="zabava">Zabava</option>
            <option value="studij">&Scaron;tudij</option>
          </select>
        </td>
      </tr>
      <tr>
        <td align="right"> </td><td><input type="submit" value="Objavi"> </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Vaja 4: CSS

Primer: vnos dnevniškega zapisa (zasnova BLOG sistema) s CSS datoteko.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Vnos v moj dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>
  <table border="0" width="100%">
   <tr><td colspan="2"><div id="glava">
               ..-N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..
                       </div></td></tr>
   <tr><td colspan="2" class="locnica"> </td></tr>
   <tr><td class="menu" width="150" valign="top"><div id="menu">Povezave</div></td>
    <td class="vsebina">
    <form method="post" action="obdelaj.php" name="prispevek">
    <table width="100%" border="0" cellpadding="2" cellspacing="2">
      <tr>
        <th align="right">Avtor prispevka</td><td> </th>
      </tr>
      <tr>
        <td align="right">Ime: </td><td><input name="ime"></td>
      </tr>
      <tr>
        <td align="right">Priimek: </td><td><input name="priimek"></td>
      </tr>
      <tr>
        <td align="right">Elektronski naslov: </td>
        <td><input name="email"></td>
      </tr>
      <tr>
        <td align="right">Naslov prispevka: </td>
        <td><input name="naslov"></td>
      </tr>
      <tr>
        <td align="right">Kategorija: </td>
        <td>
          <select name="kategorija">
            <option value="sport">&Scaron;port</option>
            <option value="politika">Politika</option>
            <option value="zabava">Zabava</option>
            <option value="studij">&Scaron;tudij</option>
          </select>
        </td>
      </tr>
      <tr>
        <th valign="top" align="right">Prispevek: </th>
        <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
      </tr>
      <tr>
        <td align="right"> </td>
        <td><input type="submit" name="submit" value="Objavi"> </td>
      </tr>
    </table>
    </form>
    </td>
   </tr>
   <tr><td colspan="2"><div id="noga">&copy; RP - Vse pravice pridržane</div></td></tr>
  </table>
</body>
</html>
```

CSS datoteka:

```
body {
font-size: 70%;
color:#000000;
background-color:#FFD833;
margin:0px;
}

body, p, h1, h2, h3, table, td, th, ul, ol, textarea, input, a {
font-family: verdana,helvetica,arial,sans-serif;
}

td.menu, td.vsebina {
background-color:#ffffff;
padding:10px;
}

td.locnica {
background-color:#CC9933;
}

div#menu {
font-size:100%;
font-weight:normal;
color:#000000;
}

div#glava {
font-size:120%;
font-variant: small-caps;
letter-spacing: 0.3cm;
font-weight:bold;
color:#660000;
margin-top:10px;
margin-bottom:5px;
text-align: center;
}

div#noga {
font-size:80%;
color:#333333;
background-color:#CC9933;
padding: 2px;
}

a:link    {color:#900B09; background-color:transparent}
a:visited {color:#900B09; background-color:transparent}
a:active  {color:#FF0000; background-color:transparent}
a:hover   {color:#FF0000; background-color:transparent; text-decoration: none;}

div#prispevek {
font-size:100%;
font-weight:normal;
color:#000000;
background-color:#FFFFCC;
margin:20px;
padding:10px;
border-style: solid;
border-color: #C0C0C0;
border-width: 1px
}
```

Vaja 5: PHP

Primer: izpišemo vnos dnevniškega zapisa

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Vnos v moj dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>
  <table border="0" width="100%">
   <tr><td colspan="2"><div id="glava">..-
N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..</div></td></tr>
   <tr><td colspan="2" class="locnica"> </td></tr>
   <tr><td class="menu" width="150" valign="top"><div id="menu">Povezave</div></td>
    <td class="vsebina">
    <?php
        echo 'Vnesli ste naslednje podatke: <br>';
        echo 'Avtor: '.$_POST["ime"].' '.$_POST["priimek"].'<br>';
        echo 'elektronski naslov: '.$_POST["email"].'<br>';
        echo 'Naslov: '.$_POST["naslov"].'<br>';
        echo 'Kategorija: '.$_POST["kategorija"].'<br>';
        echo 'Prispevek:<br>'.$_POST["prispevek"].'<br>';
    ?>
    </td>
   </tr>
   <tr><td colspan="2"><div id="noga">&copy; RP - Vse pravice pridržane</div></td></tr>
  </table>
</body>
</html>
```

Primer: preverjamo pravilnost vnosov

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Vnos v moj dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>
  <table border="0" width="100%">
   <tr><td colspan="2">
       <div id="glava">
         ..-N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..
       </div></td></tr>
   <tr><td colspan="2" class="locnica"> </td></tr>
   <tr><td class="menu" width="150" valign="top"><div id="menu">Povezave</div></td>
    <td class="vsebina">
    <?php

        if ($_POST["submit"]=="Objavi") {
           // ce smo kliknili na gumb - da kdo ne pride
           // direktno na spletno stran brez
           // ...
           // izpisovanje napak
           if (trim($_POST["ime"])=="" ||
               trim($_POST["priimek"])=="" ||
               trim($_POST["email"])=="" ||
               trim($_POST["naslov"])=="" ||
               trim($_POST["prispevek"])=="") { //ce je kaksno polje prazno
                  $error[]="Niste vnesli vseh podatkov. Vsa polja so obvezna.";
           }
           if ((strlen($_POST["ime"])<2) || (strlen($_POST["priimek"])<2)) {
                  // ce sta ime ali priimek krajsa od dveh znakov
                  $error[]="Ime ali priimek sta krajša od dveh znakov..";
```

```
            }
            if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["ime"])) {
                  $error[]="Ime vsebuje ne-alfanumerične znake.";
            }
            if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["priimek"])) {
                  $error[]="Priimek vsebuje ne-alfanumerične znake.";
            }
            if (!ereg("^[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+"."@".
                           "[-!#$%&\'*+\\/0-9=?A-Z^_`a-z{|}~]+\.".
                           "[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+$", $_POST["email"])){
                 $error[]="Elektrosnki naslov ni pravilne oblike.";
            }

            if (count($error) == 0) { // ce ni napak izpisi podatke
                echo 'Vnesli ste naslednje podatke: <br>';
                echo 'Avtor: '.$_POST["ime"].' '.$_POST["priimek"].'<br>';
                echo 'elektronski naslov: '.$_POST["email"].'<br>';
                echo 'Naslov: '.$_POST["naslov"].'<br>';
                echo 'Kategorija: '.$_POST["kategorija"].'<br>';
                echo 'Prispevek:<br>'.$_POST["prispevek"].'<br>';
           } elseif (is_array($error)) { // izpisi vse najdene napake
                echo '<p>';
                while (list($error_id,$error_text)=each($error)) {
                  echo $error_text.'<br>';
                  unset ($error[$error_id]);
                }
                echo '</p>';
            }
          } else {
            echo "Na spletno stran niste prišli preko obrazca za vnos objav.";
          }
      ?>
      </td>
     </tr>
     <tr><td colspan="2"><div id="noga">&copy; RP - Vse pravice pridržane</div></td></tr>
    </table>
</body>
</html>
```

Vaja 6: mySQL

Primer: naredimo bazo, uporabnika in tabele za naš spletni dnevnik, kamor bomo shranjevali vse zapise

```
CREATE DATABASE `rp2007`;

GRANT USAGE ON * . * TO 'rp_uporabnik'@'%' IDENTIFIED BY '******';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER,
      CREATE TEMPORARY TABLES ON `rp2007`.* TO 'rp_uporabnik'@'%';

CREATE TABLE `blog` (
     `id_blog` INT NOT NULL AUTO_INCREMENT ,
     `ime_avtor` VARCHAR( 20 ) NOT NULL ,
     `priimek_avtor` VARCHAR( 20 ) NOT NULL ,
     `e-naslov` VARCHAR( 30 ) NOT NULL ,
     `naslov` VARCHAR( 50 ) NOT NULL ,
     `datum` DATE NOT NULL ,
     `kategorija` VARCHAR( 20 ) NOT NULL ,
     `prispevek` LONGTEXT NOT NULL ,
     PRIMARY KEY ( `id` )
) TYPE = MYISAM ;

CREATE TABLE `komentar` (
     `id_komentar` INT NOT NULL AUTO_INCREMENT ,
     `id_blog` INT NOT NULL ,
     `ime` VARCHAR( 50 ) NOT NULL ,
     `e-naslov` VARCHAR( 30 ) NOT NULL ,
     `datum` DATE NOT NULL ,
     `komentar` MEDIUMTEXT NOT NULL ,
     PRIMARY KEY ( `id_komentar` )
) TYPE = MYISAM ;
```

Primer: vnesemo v bazo zapis

```
INSERT INTO `blog` ( `ime_avtor` , `priimek_avtor` , `e-naslov` ,
`naslov` , `datum` , `kategorija` , `prispevek` )
VALUES (
'Matjaž', 'Kljun', 'matjaz@pef.upr.si', 'Kam bi danes z časom',
'2007-01-29', 'zabava', 'Danes je pa en dooolgočasen dan, ko se mi
ničesar ne da narediti. Dela je pa čez glavo pa še malo več. Ne
vem, če so se v preteklosti morali ukvarjati s takimi težavami.
Verjetno ne, ker niso imeli časa misliti na neumnosti. V glavnem
ugotavljam, da je čas, v katerem živimo zelo zelo zelo
prizanesljiv do nas. Vsaj nas, ki živimo v tako imenovanem (vselej
krivično) razvitem svetu. Vprašanje je, ali bo tako tudi v naprej,
glede na vse črne napovedi, ki jih slišimo vsak dan na TV.
Vsekakor je vedno vredno optimistično sprejemati svet vase.'
);
```

Primer: preberemo iz baze vse zapise

```
SELECT * FROM `blog`;
SELECT * FROM `blog` WHERE `datum` < 2007-01-01;
```

Primer: popravimo datoteko obdelaj.php tako, da vpišemo vnos v bazo podatkov

```
 if (count($error) == 0) { // ce ni napak izpisi podatke
  echo 'Vnesli ste naslednje podatke, ki so sedaj vnešeni b zazo: <br>';
  echo '<div id="prispevek">Avtor: '.$_POST["ime"].' '.$_POST["priimek"].'<br>';
  echo 'elektronski naslov: '.$_POST["email"].'<br>';
  echo 'Naslov: '.$_POST["naslov"].'<br>';
  echo 'Kategorija: '.$_POST["kategorija"].'<br>';
  echo 'Prispevek:<br>'.$_POST["prispevek"].'<div>';

  $datum=date("Y-m-d"); //danasnji datum
  $server="localhost";
  $uporabnik="rp_uporabnik";
  $geslo="2007rp";
  $baza="rp2007";
  $db = mysql_connect($server,$uporabnik,$geslo);
  mysql_select_db($baza,$db)or die( "Unable to select database");
  //$result = mysql_query("SET NAMES utf-8",$db);
  $sql="INSERT INTO blog
        VALUES ('',
                '".$_POST["ime"]."',
                '".$_POST["priimek"]."',
                '".$_POST["email"]."',
                '".$_POST["naslov"]."',
                '$datum',
                '".$_POST["kategorija"]."',
                '".$_POST["prispevek"]."')";
   $result = mysql_query($sql,$db);
   mysql_close();
```

Primer: izdelamo index.php v katerem bomo izpisovali vse vnose.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Vnos v moj dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>
  <table border="0" width="100%">
   <tr><td colspan="2"><div id="glava">
       ..-N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..
        </div></td></tr>
   <tr><td colspan="2" class="locnica"> </td></tr>
   <tr><td class="menu" width="150" valign="top">
          <div id="menu">Povezave</div></td>
    <td class="vsebina">
         <?php

             $datum=date("Y-m-d"); //danasnji datum
             $server="localhost";
             $uporabnik="rp_uporabnik";
             $geslo="2007rp";
             $baza="rp2007";
```

```
            $db = mysql_connect($server,$uporabnik,$geslo);
            mysql_select_db($baza,$db)or die( "Unable to select database");
            //$result = mysql_query("SET NAMES utf-8",$db);

            $sql="SELECT * FROM blog ORDER BY datum LIMIT 5";
            $result = mysql_query($sql,$db);

            while ($myrow = mysql_fetch_row($result)) {
             echo '<div id="noga">'.$myrow[5].'</div>';
             echo '<div id="prispevek">
                     <a href="mailto:'.$myrow[3].'">
                       '.$myrow[1].' '.$myrow[2].'
                     </a><br>';
             echo ''.$myrow[4].'
                  <font color="grey">('.$myrow[6].')</font><br>';
             echo '<br>'.$myrow[7].'
                     </div>';
            }
         ?>
    </td>
   </tr>
   <tr><td colspan="2"><div id="noga">&copy; RP - Vse pravice
pridržane</div></td></tr>
  </table>
</body>
</html>
```

Vaja 7: racionalizacija

Primer: popravljen index.php

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

<table border="0" width="100%">
<tr><td class="menu" width="150" valign="top"><?php include('meni.php'); ?></td>
<td class="vsebina">
     <?php

     $datum=date("Y-m-d"); //danasnji datum
     $server="localhost";
     $uporabnik="rp_uporabnik";
     $geslo="2007rp";
     $baza="rp2007";
     $db = mysql_connect($server,$uporabnik,$geslo);
     mysql_select_db($baza,$db)or die( "Unable to select database");

     $sql="SELECT * FROM blog ORDER BY datum DESC LIMIT 1";
     $result = mysql_query($sql,$db);

     while ($myrow = mysql_fetch_row($result)) {
       echo '<div id="noga">'.$myrow[5].'</div>';
       echo '<div id="prispevek">
               <a href="mailto:'.$myrow[3].'">
                  '.$myrow[1].' '.$myrow[2].'
                </a><br>';
       echo ''.$myrow[4].' <font color="grey">('.$myrow[6].')</font><br>';
       echo '<br>'.$myrow[7].'
             </div>';
     }
     ?>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

Primer: glava.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type">
  <title>Naš dnevnik</title>
  <link href="blog.css" rel="stylesheet" type="text/css">
</head>
<body>
```

Primer: logo.php

```
<table border="0" width="100%">
  <tr><td><div id="glava">
       ..-N|a|š   s|p|l|e|t|n|i   d|n|e|v|n|i|k-..
      </div></td></tr>
  <tr><td class="locnica"> </td></tr>
</table>
```

Primer: meni.php

```
<div id="menu">Povezave</div>
```

Primer: noga.php

```
  <table border="0" width="100%">
   <tr><td colspan="2">
       <div id="noga">&copy; RP - Vse pravice pridržane</div></td>
   </tr>
  </table>
</body>
</html>
```

Primer: datoteka primer.html smo preimenovali v vnos.php

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
    <td class="vsebina">
    <form method="post" action="obdelaj.php" name="prispevek">
    <table width="100%" border="0" cellpadding="2" cellspacing="2">
      <tr>
        <th align="right">Avtor prispevka</td><td> </th>
      </tr>
      <tr>
        <td align="right">Ime: </td><td><input name="ime"></td>
      </tr>
      <tr>
        <td align="right">Priimek: </td><td><input name="priimek"></td>
      </tr>
      <tr>
        <td align="right">Elektronski naslov: </td>
        <td><input name="email"></td>
      </tr>
      <tr>
        <td align="right">Naslov prispevka: </td>
        <td><input name="naslov"></td>
      </tr>
      <tr>
        <td align="right">Kategorija: </td>
        <td>
          <select name="kategorija">
            <option value="sport">&Scaron;port</option>
            <option value="politika">Politika</option>
            <option value="zabava">Zabava</option>
            <option value="studij">&Scaron;tudij</option>
          </select>
        </td>
      </tr>
      <tr>
        <th valign="top" align="right">Prispevek: </th>
        <td><textarea cols="25" rows="10" name="prispevek"></textarea></td>
      </tr>
      <tr>
        <td align="right"> </td>
        <td><input type="submit" name="submit" value="Objavi"> </td>
      </tr>
```

```
    </table>
    </form>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

Primer: obdelaj.php

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
         <?php include('meni.php'); ?></td>
    <td class="vsebina">
    <?php

        if ($_POST["submit"]=="Objavi") {
           // ce smo kliknili na gumb - da kdo ne pride
           // direktno na spletno stran brez
           //izpisovanje napak
           if (trim($_POST["ime"])=="" ||
               trim($_POST["priimek"])=="" ||
               trim($_POST["email"])=="" ||
               trim($_POST["naslov"])=="" ||
               trim($_POST["prispevek"])=="") { //ce je kaksno polje prazno
                  $error[]="Niste vnesli vseh podatkov. Vsa polja so obvezna.";
           }
           if ((strlen($_POST["ime"])<2) || (strlen($_POST["priimek"])<2)) {
                  // ce sta ime ali priimek krajsa od dveh znakov
                  $error[]="Ime ali priimek sta krajša od dveh znakov..";
           }
           if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["ime"])) {
                  $error[]="Ime vsebuje ne-alfanumerične znake.";
           }
           if (!ereg("([A-Z]{1})[[:alpha:]]", $_POST["priimek"])) {
                  $error[]="Priimek vsebuje ne-alfanumerične znake.";
           }
           if (!ereg("^[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+"."@".
                    "[-!#$%&\'*+\\/0-9=?A-Z^_`a-z{|}~]+\.".
                    "[-!#$%&\'*+\\./0-9=?A-Z^_`a-z{|}~]+$", $_POST["email"])){
                 $error[]="Elektrosnki naslov ni pravilne oblike.";
           }

           if (count($error) == 0) { // ce ni napak izpisi podatke
              echo 'Vnesli ste naslednje podatke: <br>';
              echo '<div id="prispevek">Avtor:
                   '.$_POST["ime"].' '.$_POST["priimek"].'<br>';
              echo 'elektronski naslov: '.$_POST["email"].'<br>';
              echo 'Naslov: '.$_POST["naslov"].'<br>';
              echo 'Kategorija: '.$_POST["kategorija"].'<br>';
              echo 'Prispevek:<br>'.$_POST["prispevek"].'<div>';

              $datum=date("Y-m-d"); //danasnji datum
              $server="localhost";
              $uporabnik="rp_uporabnik";
              $geslo="2007rp";
              $baza="rp2007";
              $db = mysql_connect($server,$uporabnik,$geslo);
              mysql_select_db($baza,$db)or die( "Unable to select database");
```

```
            $sql="INSERT INTO blog
                    VALUES ('','".$_POST["ime"]."',
                            '".$_POST["priimek"]."',
                            '".$_POST["email"]."',
                            '".$_POST["naslov"]."',
                            '$datum',
                            '".$_POST["kategorija"]."',
                            '".$_POST["prispevek"]."')";
              echo $sql;
              $result = mysql_query($sql,$db);
              mysql_close();

          } elseif (is_array($error)) { // izpisi vse najdene napake
              echo '<p>';
              while (list($error_id,$error_text)=each($error)) {
                echo $error_text.'<br>';
                unset ($error[$error_id]);
              }
              echo '</p>';
          }
        } else {
          echo "Na spletno stran niste prišli preko obrazca
                za vnos objav v dnevniški sistem";
        }
    ?>
    </td>
   </tr>
  </table>

<?php include('noga.php'); ?>
```

Vaja 8: Zaključek projekta

Primer: popravljen meni.php

```
<div id="menu">

<a href="index.php">Moj blog</a><br>
<a href="vnos.php">Vnos prispevka</a><br><br>


<?php

$month=date("m");
$year=date("Y");

for($j=$year;$j>($year-3);$j--){
  echo '<hr><p>'.$j.'</p><p align="center">';
  if ($j==$year) {
    for ($i=$month; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  } else {
    for ($i=12; $i>0 ; $i--) {
      //posiljanje spletni strani podatke preko URL - GET metoda
      echo '|-<a href="index.php?year='.$j.'&month='.$i.'">'.$i.'</a>-';
      if($i==9||$i==5) echo '|<br>';
    }
    echo '|';
  }
  echo '</p><hr>';
}
?>

</div>
```

Primer: popravljen index.php (dodan obrazec za komentarje, vnos komentarja v bazo, izpis komentarjev pri posameznem vnosu in izpis vnosov enega mesca)

```
<?php include('glava.php'); ?>
<?php include('logo.php'); ?>

  <table border="0" width="100%">
   <tr><td class="menu" width="150" valign="top">
       <?php include('meni.php'); ?></td>
    <td class="vsebina" valign="top">
      <?php

      $server="localhost";
      $uporabnik="rp_uporabnik";
      $geslo="2007rp";
      $baza="rp2007";
      $db = mysql_connect($server,$uporabnik,$geslo);
      mysql_select_db($baza,$db)or die( "Unable to select database");

      if ($_POST["submit"] && $_POST["id_blog"]) {
        //vnesi komentar, če kdo klikne na gumb
        $datum=date("Y-m-d"); //danasnji datum
        $sql="INSERT INTO komentar
              VALUES ('','".$_POST["id_blog"]."',
```

```
                    '".$_POST["ime"]."',
                    '".$_POST["email"]."',
                    '$datum',
                    '".$_POST["komentar"]."')";
      $result = mysql_query($sql,$db);
    }

    //kaj izpisemo je odvisno od kod prihajamo (na kaj smo kliknili)
    if ($_POST["id_blog"]) { // ce je poslan id zapisa ga najdi
      $sql="SELECT * FROM blog WHERE id_blog=".$_POST["id_blog"];
    } elseif ($_GET["id_blog"]) {// ce je poslan id zapisa ga najdi
      $sql="SELECT * FROM blog WHERE id_blog=".$_GET["id_blog"];
    } elseif ($_GET["month"]&&$_GET["year"]) { // ce imamo mesec in leto
      //najdemo vse zapise tega leta in meseca
      $sql="SELECT * FROM blog WHERE datum
            LIKE '".$_GET["year"]."-".$_GET["month"]."%'";
    } else { // ce ne pa najdi zadnjega zapisanega
      $sql="SELECT * FROM blog ORDER BY datum DESC LIMIT 1";
      $prazen=1; // tega potrebujemo za izpis komentarjev zadnjega
    }

    $result = mysql_query($sql,$db);
    while ($myrow = mysql_fetch_row($result)) { //dokler je se kaj najdenega
      echo '<div id="noga">'.$myrow[5].'</div>';
      echo '<div id="prispevek">
        <a href="mailto:'.$myrow[3].'">'.$myrow[1].' '.$myrow[2].'</a><br>';
      echo '<a href="index.php?id_blog='.$myrow[0].'">'.$myrow[4].'</a>
        <font color="grey">('.$myrow[6].')</font><br>';
      echo '<br>'.$myrow[7].'
        </div>';

    if ($_POST["id_blog"] || $_GET["id_blog"] || $prazen==1) {
      // izpis komentarjev
      $db2 = mysql_connect($server,$uporabnik,$geslo);
      mysql_select_db($baza,$db2)or die( "Unable to select database");
      $sql2="SELECT * FROM komentar WHERE id_blog=".$myrow[0];
      $result2 = mysql_query($sql2,$db2);
      while ($myrow2 = mysql_fetch_row($result2)) {
         echo '<hr><p><a href="mailto:'.$myrow2[3].'">'.$myrow2[2].'</a>
               ('.$myrow2[4].'): '.$myrow2[5].'</p>';
      }
      // obrazec za vpis komentarja
      echo '<hr>
           <form method="post" action="index.php">
           <input type="hidden" name="id_blog" value="'.$myrow[0].'">
           Ime: <input name="ime"> <br>
           E-naslov: <input name="email"> <br>
           Komentar: <br>
           <textarea cols="50" rows="4" name="komentar"></textarea><br>
           <input type="submit" name="submit" value="Objavi">';
    }
   }

   mysql_close();
   ?>
   </td>
  </tr>
 </table>

<?php include('noga.php'); ?>
```

Vaja 10: SED

Sed je ukaz, s katerim obdelujemo besedilo. Iz datotek ali iz drugega vhoda (tipkovnica ali izhod drugega programa). Ukazi se izvajajo po vrsticah. Če imamo več vrstično datoteko se ukaz izvede na vsaki vrstici posebej. Izhod ukaza sed je ekran. Sed ne spreminja datotek.

```
$ sed -e 'd' /etc/services
```

Sed odpre /etc/services in naloži vsako vrstico posebej v svoj „notranji spomin“. Ko je vrstica v „notranjem spominu“ izvede nad njo ukaz – v našem primeru zbriše vsako vrstico posebej (d). Na izhodu izpiše zbrisane vrstice; torej nič. Datoteka ni bila spremenjena, ker jo sed uporablja samo za vhod. Ukaze pišemo v narekovajih.

```
$ sed -e '1d' /etc/services | more
```

Ukaz zbriše iz iste datoteke samo prvo vrstico. Številka torej pomeni številko vrstice.

```
$ sed -e '1,10d' /etc/services | more
```

Ukaz zbriše prvih 10 vrstic iz datoteke.

Kaj pa regularni izrazi? Poglejmo kako lahko izpišemo iz iste datoteke vse vrstice razen tistih v katerih so komentarji (te se začnejo z znakom #).

```
$ sed -e '/^#/d' /etc/services | more
```

'/^#/' - regularne izraze v sed ukazu vedno damo v poševnice.

Ponovimo na kratko regularne izraze:

Character	**Description**
^	začetek vrstice
$	konec vrstice
.	kakšen koli znak
*	nič ali več ponovitev predhodnega znaka
[ ]	vsi znaki znotraj [ ]

Primeri	**Opis**
/./	vse vrstice s samo enim znakom
/../	vse vrstice z dvema znakoma
/^#/	vse vrstice, ki se začnejo z znakom #
/^$/	vse prazne vrstice
/}^/	vse vrstice ki se konlajo z znakom }
/} *^/	vse vrstice, ki se končajo z } in niča li več presledki

/[abc]/ vse vrstice ki vsebujejo a, b, c
/^[abc]/ vse vrstice, ki se začnejo z a, b ali c

Regularne izraze torej uporabljamo na sledeč način

```
$ sed -e '/regexp/d' /path/to/my/test/file | more
```

To bo vse vrstice, razen tistih, ki ustrezajo regularnemu izrazu. Obraten ukaz, ki izpiše vse vrstice, ki ustrezajo regularnemu izrazu dobimo z zamenjavo d s p. Torej .n nam pove, naj nič ne izpišemo na izhod, medtem ko p eksplicitno zahteva izpisovanje.

```
$ sed -n -e '/regexp/p' /path/to/my/test/file | more
```

Kaj pa če želimo izpisati vse vrstice med dvema vrsticama, ki ustrezata regularnima izrazoma.

```
$ sed -n -e '/BEGIN/,/END/p' /my/test/file | more
```

Regularna izraza ločimo med seboj z vejico. Če nobena vrstica ne ustreza prvemu izrazu se ne izpiše nič. Če ena vrstica ustreza prvemu in nobena drugemu izrazu se izpiše vse do konca datoteke.

Če želimo izpisati samo main funkcijo iz c programa uporabimo naslednji ukaz:

```
$ sed -n -e '/main[[:space:]]*(/,/^}/p' sourcefile.c | more
```

Dva regularna izraza: '/main[[:space:]]*(/' in '/^}/'. Seveda to izpiše le, če je vaša koda lepo oblikovana. Torej če se main funkcija zakljuli z vrstico, ki se začne z { znakom.

```
$ sed 's/UP PEF/Pedagoska fakulteta koper/g' datoteka.txt
```

Ukaza s in g pa zamenjata prvi niz z drugim.

Vaja 11: AWK

Programi ukazne vrstice

Imamo datoteko kovanci.txt z nalslednjimi stoplci:

```
kovina teža   leto_izkopa  drzava_izvora        opis
```

Vsebina datoteke je naslednja – v vsaki vrstici imamo zabeležene podatke enega kovanca:

```
gold     1    1986          USA                  American Eagle
gold     1    1908          Austria-Hungary      Franz Josef 100 Korona
silver  10    1981          USA                  ingot
gold     1    1984          Switzerland          ingot
gold     1    1979          RSA                  Krugerrand
gold     0.5  1981          RSA                  Krugerrand
gold     0.1  1986          PRC                  Panda
silver   1    1986          USA                  Liberty dollar
gold     0.25 1986          USA                  Liberty 5-dollar piece
silver   0.5  1986          USA                  Liberty 50-cent piece
silver   1    1987          USA                  Constitution dollar
gold     0.25 1987          USA                  Constitution 5-dollar piece
gold     1    1988          Canada               Maple Leaf
```

Izpišemo vse vrstice, ki vsebujejo besedo gold:

```
awk '/gold/' kovanci.txt
```

Enako bi lahko izvedli z ukazi grep ali find. Vendar je to le osnovni primer. AWK zmore veliko več. Recimo, da želimo izpisati samo opis vseh vrstic:

```
awk '/gold/ {print $5,$6,$7,$8}' kovanci.txt
```

Program sedaj izpiše samo opise vrstic, ki vsebujejo besedo gold.

Osnovna struktura ukaza je:

```
awk <iskalni niz> {<akcije>}
```

Iskalne nize tvorimo podobni kot pri SED. Lahko seveda uporabimo regularne izraze.
Ker imamo v naši datoteki največ 8 nizov, ločenih s presledki ali tabulatorji, lahko zapišemo, kateri stolpec ustreza določenemu opisu:

```
kovina:        $1
teža:          $2
leto_izkopa:   $3
drzava_izvora: $4
opis:          $5 do $8
```

AWKova privzeta akcija je izpis cele vrstice. Spodnji primeri naredijo enako:

```
awk '/gold/'
awk '/gold/ {print}'
awk '/gold/ {print $0}'
```

Kontrola toka.

Izpisati želiomo vse vrstice pri katerih je bil izkop izveden pred letom 1980:

```
awk '{if ($3 < 1980) print $3, "    ",$5,$6,$7,$8}' kovanci.txt
```

Pri tem primeru lahko vidimo naslednje:

- ker ni iskalnega niza AWK prebere vse vrstice in na vseh izvede akcijo
- izpisu print lahko dodamo lastno besedilo z dvojnimi narekovaji
- print se izvrši le, če je if pogoj resničen
- enako kot pri PHP AWK ne razlikuje med števili in besedami; če niz vsebuje število lahko nad njim izvajamo aritmetične operacije.

Preštejmo število vseh kovancev, kar pomeni izpis števila vrstic:

```
awk 'END {print NR,"coins"}' kovanci.txt
```

Izpis je sledeč:

```
13 coins
```

Razširjena oblika ukaza je:

```
awk 'BEGIN               {<initializations>}
     <search pattern 1> {<program actions>}
     <search pattern 2> {<program actions>}
     ...
     END                 {<final actions>}'
```

Poleg stavka BEGIN se izvede še inicializacija. Nato sledijo iskalni nizi po vrsticah in nad vsakim izvedemo določeno akcijo. AWK pregleda vsako prebrano vrstico datoteke, če morda ustreza kakšnemu iskalnemu nizu. Na koncu END poleg katerega lahko napišemo še dodatne dogodke, ko smo datiteko že prebrali.

V našem primeru torej v sami datoteki s podatki ne storimo nilesar ampak samo izvedemo zadnjo akcijo, ki izpiše število vrstic v NR (number of records) spremenljivki:

Naslednji primer nam bo izračunal vrednost vseh zlatnikov. Predpostavimo, da je teža zlata kovanca po enoti teže približno 400€. Vrednost vseh zlatnikov je torej:

```
awk '/gold/ {teza += $2} END {print "value = " 425*teza "€"}' kovanci.txt
```

Kar izpiše:

```
value = $2592.5
```

Spremenljivko teza smo začeli uporabljati sredi programa. Imena spremenljivk niso omejena, če le niso rezervirana ("print", "NR", "END", ...). Spremenljivke si na začetku prazne (null za niz on 0 za število).

V vsaki vrstici, kjer iamo besedo gold se spremenljivnki teza pristeje drugi stolpec. Kar bi lahko zapisali tudi:

```
{ounces = ounces + $2}
```

AWK Programi

Kaj pa če želimo izvesti nad datoteko kovanci.txt več akcij hkrati. Namesto da vse ukaze vpišemo v ukazno vrstico, jih lahko zapišemo v datoteko, ki jo kličemo na sledeč način:

```
awk -f <awk program file name>
```

Predpostavimo da želimo spedeči izpis:

```
Povzetek podatkov naše zbirke kovancev:
```

```
Število zlatnikov:      nn
Teža zlatnikov:         nn.nn
Vrednost zlatnikov:     n,nnn.nn

Število srebrnikov:     nn
Teža srebrnikov:        nn.nn
Vrednost srebrnikov:    n,nnn.nn

Število vseh kovancev:  nn
Vrednost zbirke:        n,nnn.nn
```

Naslednji AWK program izpiše zgornji izpis:

```
# This is an awk program that summarizes a coin collection.
#
/gold/    { num_gold++; wt_gold += $2 }      # Teza zlata.
/silver/  { num_silver++; wt_silver += $2 }  # Teza srebra.
END { val_gold = 485 * wt_gold;              # Vrednost zlata.
      val_silver = 16 * wt_silver;           # Vrednost srebra.
      total = val_gold + val_silver;
      print "Povzetek podatkov naše zbirke kovancev:";  # Tiskaj.
      printf ("\n");
      printf ("   Število zlatnikov:                  %2d\n", num_gold);
      printf ("   Teža zlatnikov:          %5.2f\n", wt_gold);
      printf ("   Vrednost zlatnikov:        %7.2f\n",val_gold);
      printf ("\n");
      printf ("   Število srebrnikov:                %2d\n", num_silver);
      printf ("   Teža srebrnikov:        %5.2f\n", wt_silver);
      printf ("   Vrednost srebrnikov s:      %7.2f\n",val_silver);
      printf ("\n");
      printf ("   Število vseh kovancev:        %2d\n", NR);
      printf ("   Vrednost zbirke v €:         %7.2f\n", total); }
```

* I stored this program in a file named "summary.awk", and invoked it as follows:

```
awk -f povzetek.awk kovanci.txt
```

Vaja 12: Programiranje v lupini

Lupinska skripta (ekvivalent datoteki .BAT v MSDOS-u) je enostavna tekstovna datoteka z ukazi, ki jo lahko napišemo s katerimkoli urejevalnikom (vi, emacs,…), shranimo in napravimo izvedljivo z ukazom chmod +x ime_skripte.

Končnica datoteke je odvisna od lupine v kateri programiramo. Najpodostejša sta sh in bash
Skripto zaženemo z ukazom: sh ime_skripte.sh ali ,/ime_skripte.sh
Prva vrstica skripte običajno vsebuje pot do lupinskega interpreterja: #!/bin/sh
Komentarje začnemo z znakom #

Izpisi

```
#!/bin/sh
#
clear # zbrišemo zaslon
echo "Pozdravljen `whoami`" # uporabimo zunanji ukaz za
# izpis trenutnega uporabnika
echo "Danes je `date`" # izpiše datum
```

Spremenljivke in aritmetika

```
#!/bin/sh
#
ime=Janez
echo "Moje ime je $ime"
stev1=5
stev2=10
rezultat1=`expr $stev1 + $stev2` # izračun vsote števil
echo "Vsota je $rezultat1" # izpis rezultata
# še ostale aritmetične operacije
rezultat2=`expr $stev1 / $stev2` # celoštevilčno deljenje
rezultat3=`expr $stev1 % $stev2` # ostanek pri deljenju
rezultat4=`expr $stev1 \* $stev2` # množenje števil
```

Vnos vrednosti preko tipkovnice

```
#!/bin/sh
#
echo "Vpiši svoje ime:"
read ime # prebere niz iz tipkovnice
echo "Tvoje ime je $ime" # izpiše niz
```

Vnos vrednosti s pomočjo parametrov ukazne vrstice

```
#!/bin/sh
#
echo "Tej skripti je ime $0" # vgrajene spremenljivke
echo "Prvi parameter je: "$1
echo "Vse skupaj je $# parametrov, in sicer: "$*
```

Stavek if

```
#!/bin/sh
#
if [ $# -lt 1 ]
then
```

```
echo "Zagon: ime_skripte ime_datoteke"
exit 1
fi
#
if rm $1 # če uspe brisanje datoteke
then # potem izpišemo, da smo jo izbrisali
echo "datoteka $1 je bila izbrisana"
fi # konec if stavka
```

Stavek test

```
#!/bin/sh
# skripta ugotovi ali je argument pozitiven
#
if test $1 -gt 0 # testiramo ali je število > 0
then
echo "$1 število je pozitivno"
fi
```

Stavek for

```
#!/bin/sh
# izpiše števila od 2 do 5
#
for i in 2 3 4 5
do
echo " Številka $i"
done # zaključimo zanko for
#!/bin/sh
# izpiše števila od 0 do 5
#
for ((i=0; i<=5; i++))
do
echo "Številka $i"
done
#!/bin/sh
# skripta prešteje število vseh datotek in
# število vseh podimenikov
#
DIRS=0 # število imenikov
FILES=0 # število datotek
#
for file in ´ls .´ # uporabimo zunanji ukaz ls .
do
if [ -d ${file} ]
then # -d imenik?
DIRS=´expr $DIRS + 1´ # DIRS=DIRS+1
elif [ -f ${file} ]
then # drugače -f datoteka?
FILES=´expr $FILES + 1´ # FILES=FILES+1
fi # konec if stavka
done # zaključimo zanko for
#
echo "Imenikov je ${DIRS}, datotek pa ${FILES}"
```

Stavek case

```
#!/bin/sh
# skripta glede na pripono datoteke ugotovi ali ta
# vsebuje sliko, besedilo, izvorni program ali drugo
#
for file in ´ls .´ # uporabimo zunanji ukaz ls .
do
case ${file} in
*.gif|*.jpg) echo "${file} slika";;
*.txt) echo "${file} besedilo";;
*.c|*.f) echo "${file} izvorna koda programa";;
*) echo "${file} druga datoteka";;
esac # zaključimo stavek case
done # zaključimo zanko for
```