# Osnove računarstva II

Script fajlovi i naredbe za kontrolu toka programa

# Scrip fajlovi

- Komande se mogu grupisati u jedan tekstualni fajl i izvršavati po potrebi.
- Takav fajl nazivamo m-fajlom ili script fajlom, ekstenzija mu je .m.
- m-fajlove možemo kreirati u bilo kom tekstualnom editoru.
- Sadržaj fajlova su komande koje se izvršavaju sekvencijalno (jedna po jedna, redom kojim su navedene).
- Razvojno okruženje je program namijenjen pisanju koda, editovanju i ispravljanju grešaka (debug-ovanje).
- Postoje integrisana razvojna okruženja (eng. integrated development environment - IDE) koja omogućavaju i testiranje programa, tj. koja uključuju programski prevodilac.
- Programski prevodilac je program koji omogućava da se program napisan u višem programskom jeziku (tzv. izvorni kôd) prevede u jezik 0 i 1, razumljiv računaru (tzv. mašinski kôd).

# Programski prevodioci – interpreteri

- Programski prevodioci se još nazivaju translatori.
- Dvije osnovne grupe translatora su interpreteri i kompajleri.
- Interpreteri prolaze kroz izvorni kôd naredbu po naredbu, vrše prepoznavanje naredbe i pozivaju dio koda intepretera koji tumači tu naredbu.
- Prednosti interpretera:
  - relativno su jednostavni,
  - ne stvaraju izvršne verzije programa na računaru, pa štede memoriju.
- Mane interpretera:
  - sporost,
  - program se ne može tumačiti bez interpretera,
  - ne postoji mogućnost sakrivanja sopstvene programerske vještine.
- Poznati interpreteri su Basic, Java, Octave/MATLAB.
- U novijim verzijama interpretera postoji mogućnost sakrivanja koda.

# Programski prevodioci – kompajleri

- Kompajleri prave izvršnu verziju programa (.exe fajl), koji predstavlja samostalnu aplikaciju (može se izvršavati bez dodatnog softvera).
- Kompajliranje je znatno složenije od interpretiranja.
- Prednosti kompajlera:
  - brzina izvršavanja,
  - sakrivanje programerske vještine,
  - finalni produkt je samostalan, bez potrebe za isporučivanjem programa za tumačenje (kompajliranje).
- Mana kompajlera je složenost. Kompajleri su skupi programi. Kreiranje jednog kompajlera je vrlo složen proces. Detaljno izučavanje procesa kompajliranja po pravilu zahtijeva poseban kurs.
- Jezici čiji se programi kompajliraju su C, C++, Fortran, Pascal, Visual Basic.

# Scrip fajlovi

- Fajl „startujemo“ navođenjem njegovog imena (bez .m) u komandnoj liniji Octave/MATLAB-a, pri čemu vodimo računa da se nalazimo u istom folderu u kojem je snimljen fajl.
- Ime fajla mora poštovati ista pravila kao ime varijable (mora početi slovom, sastoji se od slova engleskog alfabeta, cifara i podvlake).
- **Nije dozvoljen razmak u nazivu m-fajla.**
- Script fajlovi olakšavaju ispravljanje grešaka i doradu koda.
- **Nemaju parametre pri pozivu (samo ime fajla).**
- Vrijednosti promjenljivih ostaju u radnom okruženju nakon izvršavanja m-fajla i naredne naredbe ih mogu koristiti kao poznate.

# Scrip fajlovi

- m-fajlovi rade sa aktivnim radnim prostorom, odnosno u njima možemo koristiti sve varijable koje su definisane u radnom prostoru prije startovanja m-fajla.
- Promjenljive se mogu unijeti u toku izvršavanja programa korišćenjem komande `a = input('zeljeni tekst');`
- Nakon naredbe `input` ispisuje se u komandnom prozoru željeni tekst i čeka se unos vrijednosti sa tastature, koja se dodjeljuje promjenljivoj `a`.
- Ukoliko se ne navede promjenljiva u koju se treba smjestiti unos, isti se smješta u promjenljivu `ans`.

# Scrip fajlovi – interakcija sa korisnikom

- `keyboard` – privremeno obustavlja izvršavanje programa i vraća nas na komandni prozor i tastaturu.
- `return` – vraća kontrolu programu i nastavlja sa izvršavanjem programa. `dbcont` posle MATLAB 2015b i u Octave
- `pause` – privremeno zaustavlja izvršavanje programa.
- `pause(n)` – pravi pauzu od *n* sekundi.
- `pause off` – onemogućava zaustavljanje računara u pause modu.
- `pause on` – omogućava zaustavljanje računara u pause modu.
- `echo` – omogućava praćenje izvršavanja programa.
- `diary ime.tip` – smješta sve što je kucano ili ispisivano u komandnoj liniji u fajl ime.tip.
- `diary off` - vrši suspenziju diary moda, nakon čega se može pristupiti kreiranom fajlu.

# Primjer

- Nacrtati grafik funkcije $f(x) = x^2 \sin(x + \varphi)$, $-\pi \leq x \leq \pi$, a zatim izračunati integral u istom intervalu koristeći 40 odbiraka i nacrtati neodređeni integral u istom intervalu.

```
dx = 2*pi/40;
x = -pi : dx : pi-dx;
keyboard
fx = x.^2 .* sin(x+phi);
plot(x, fx)
hold on
pause
in = sum(fx) * dx
pause
in2 = cumsum(fx) * dx;
plot(x + dx, in2,'r')
```

Nakon naredbe keyboard kontrola se prenosi na tastaturu i prihvataju se komande koje se kucaju u komandnom prozoru. To ćemo iskoristiti da bismo dodijelili vrijednost promjenljivoj phi, npr:

```
K>> phi = pi / 2
phi =
    1.5708
K>> dbcont %vraća komandu programu
```

Pauzira dalje izvršavanje programa, sve dok se ne pritisne bilo koja tipka

# Kontrola toka programa – selekcija

- Uslovno izvršavanje dijela programa.
- Sintaksa naredbe:

```
if (uslov)
    naredbe
end
```

Ključna reč `end` označava kraj `if` naredbe

- Ukoliko je logički uslov ispunjen izvršavaju se naredbe, u suprotnom se ide na prvu naredbu posle end.
- Primjer:

```
n = input('Unesi broj ');
if (n > 10)
   disp('n je vece od 10');
end
```

`input` funkcija za unos podatka

`disp` funkcija za ispis

# Logički i operatori poređenja

- Logičke i operatore poređenja ćemo koristiti za formiranje složenih logičkih uslova

| Operator | Operacija |
|---|---|
| && | Logičko I |
| \|\| | Logičko ILI |
| ~ | Logička negacija |

| Operator | Operacija |
|---|---|
| > | Veće od |
| >= | Veće od ili jednako |
| < | Manje od |
| <= | Manje od ili jednako |
| == | Provjera jednakosti |
| ~= | Provjera nejednakosti |

# Kontrola toka programa – selekcija

- Sintaksa naredbe:

```
if (uslov)
   naredbe1
else
   naredbe2
endif - end u MATLAB-u
```

Zagrada nije obavezna

- Ukoliko je logički uslov ispunjen izvršavaju se `naredbe1`, u suprotnom se izvršavaju `naredbe2`.

- Primjer:

```
n = input('Unesi broj ');
if (n > 10)
  disp('n je vece od 10');
else
  disp('n je manje ili jednako 10');
end % funkcioniše, ali neće automatski poravnati kod
```

U else grani nema uslova!!!
Česta greška je pisanje uslova.

# Kontrola toka programa – selekcija

- Sintaksa naredbe:

```
if (uslov1)
   naredbe1
elseif (uslov2)
   naredbe2
...
else
   naredbeK
endif
```

Uslovi se provjeravaju redom i izvršavaju se prve naredbe čiji je uslov ispunjen. Ukoliko nijedan uslov nije ispunjen, izvršavaju se naredbe u else grani.
Nakon izvršavanja odgovarajućih naredbi, nastavlja se sa naredbama nakon ključne riječi end.

- Primjer:

```
n = input('Unesi broj ');
if (n > 10)
   disp('Broj je veci od 10');
elseif (n < 3)
   disp('Broj je manji od 3');
else
   disp('Broj je izmedju 3 i 10');
endif
```

# Programske petlje – for petlja

- U Octave/MATLAB-u mogu da se koriste **for** i **while** petlje.
- for petlja (ili brojačka petlja) se koristi kada se zna koliko puta treba da se izvrši određeni dio koda (tijelo petlje).
- Sintaksa for petlje:

  **for i = a : b : c**

  **naredbe**

  **endfor (end u MATLAB-u)**

  gdje je:

  i – brojačka promjenljiva

  a – početna vrijednost brojača *i*

  b – korak promjene brojača *i*

  c – krajnja vrijednost brojača *i*

Primjer

```
for i = 1 : 2 : 10
   disp(i)
endfor
     1
     3
     5
     7
     9
```

# Primjer sa for petljom

- Napisati m-fajl **Niz** kojim se unosi niz **X** i broj **N** i koji određuje i štampa koliko se puta broj N pojavljuje u nizu X.

```
X = input('Unijeti niz X ');
N = input('Unijeti broj N ');
Br = 0;
for i = 1 : length(X)
   if X(i) == N
      Br = Br + 1;
   endif
endfor
if Br == 0
   disp('Nema ga!');
else
   disp('Broj pojava je');
   disp(Br);
endif
```

Jedno izvršenje

```
>> Niz
Unijeti niz X [1 2 5 4 2 5 1]
Unijeti broj N 5
Broj pojava je
     2
```

# Primjer sa for petljom

- Formirati matricu dimenzija *NxN*, *N* unosi korisnik po sljedećem pravilu:

$$a(i,j) = \begin{cases} i-j, i \neq j, i < j \\ -a(j,i), i > j \\ i^2, i = j \end{cases}$$

```
N = input('Unesite dimenzije matrice ');
a = zeros(N);
for i = 1 : N
    for j = 1 : N
        if i ~= j & i < j
            a(i,j) = i - j;
        elseif i > j
            a(i,j) = -a(j,i);
        else
            a(i,j) = i^2;
        endif
    endfor
endfor
disp(a)
```

Jedno izvršenje

```
>> primjer2
Unesite dimenzije matrice 5
     1    -1    -2    -3    -4
     1     4    -1    -2    -3
     2     1     9    -1    -2
     3     2     1    16    -1
     4     3     2     1    25
```

# while petlja

- **while** petlja se koristi kada se unaprijed ne zna koliko puta treba da se izvrši određeni dio koda, ali se zna uslov do kada treba da se izvršava.

- Sintaksa while petlje:

```
while uslov
    naredbe
endwhile
```

gdje je uslov logički uslov koji određuje do kad se izvršava petlja.

```
i = 1
while i < 10
   disp(i)
   i = i + 3
endwhile
```

```
1
4
7
```

# Primjer sa while petljom

- Data je jednakost: $$\pi^2 = \sum_{n=1}^{\infty} \frac{6}{n^2}$$

Napisati m-fajl pod nazivom piKvadrat koji približno računa vrijednost $\pi^2$ koristeći datu sumu. Sumiranje prekinuti kad razlika približne i tačne vrijednosti postane manja od $10^{-5}$. Na izlazu ispisati dobijenu vrijednost sume.

```
eps = 1e-5; %definišemo grešku računanja, zasjenili smo
n = 1; suma = 0; % ugradjeno eps
while abs(pi^2-suma) > eps
    suma = suma + 6 / n ^ 2;
    n = n + 1;
endwhile
disp('Dobijena vrijednost sume je ');
disp(suma);
```

Jedno izvršenje

```
>> piKvadrat
Dobijena vrijednost sume je
    9.8696
```

# Naredba break

- Izvršenje for i while petlje se može prekinuti naredbom **break**. Nakon naredbe break, prelazi se na prvu naredbu nakon petlje.

- Napisati m-fajl saberiDoPrveNule koji za unijeti niz brojeva X sumira elemente tog niza sve dok ne naiđe na 0 i ispisuje dobijenu sumu i broj sumiranih elemenata. (Odraditi kod kuće zadatak uz pomoć while petlje)

```
clear all %Dobro je na pocetku obrisati radnu memoriju
X = input('Unesi niz X');
suma = 0;broj = 0;
for i = 1 : length(X)
    if(X(i) == 0)
        break
    else
        suma = suma + X(i);
        broj = broj + 1;
    endif
endfor
disp('Broj sumiranih elemenata je')
disp(broj)
disp('Suma je')
disp(suma)
```

Jedno izvršenje

```
>> saberiDoPrveNule
Unesi niz X[1 2 5 3 0 2 5]
Broj sumiranih elemenata je
     4
Suma je
    11
```

# Naredba error

- Komandom **error**(’Neki tekst’) prekidamo izvršenje programa i korisniku prikazujemo tekst greške.

```
x = input('Unesi niz'); n = input('Unesi broj');
if(n >= 0 & n <=9 & fix(n)==n)
    suma = 0;
    for i = 1 : length(x)
        pom = x(i); cif = 0;
        while(x(i) > 0)
            cif = cif + 1;
            x(i) = fix(x(i) / 10);
        end
        if(cif + n == 10)
            suma = suma + pom;
        end
    endfor
    suma
else
    error('Nijeste unijeli cifru')
endif
```

Kreirati m-fajl sedmi kojim se unosi niz cijelih brojeva **x** i cio broj *n*. Algoritam na izlazu daje sumu onih elemenata niza čiji **broj** cifara sabran sa *n* daje broj 10. Ispitati nakon unošenja, da li je unijeti broj *n* cifra.

Jedno izvršenje

```
>> sedmi

Unesi niz[235 54 2145 365]
Unesi broj7
suma =  600
```