Predavanje 5 - Programiranje u MATLABu

Tatjana Tomović

Institut za matematiku i informatiku
Prirodno-matematički fakultet
Univerzitet u Kragujevcu

Kragujevac, 2014.

Pregled predavanja

M fajlovi

- Komanda *edit ime_fajla* otvara prozor editora m fajlova i priprema *file ime_fajla* za editovanje.

M fajlovi

- Komanda *edit ime_fajla* otvara prozor editora m fajlova i priprema *file ime_fajla* za editovanje.
- Skript fajlovi se koriste kad je neophodno neki niz komandi Matlaba ponavljati mnogo puta.

M fajlovi

- Komanda *edit ime_fajla* otvara prozor editora m fajlova i priprema *file ime_fajla* za editovanje.
- Skript fajlovi se koriste kad je neophodno neki niz komandi Matlaba ponavljati mnogo puta.

M fajlovi

- Komanda *edit ime_fajla* otvara prozor editora m fajlova i priprema *file ime_fajla* za editovanje.
- Skript fajlovi se koriste kad je neophodno neki niz komandi Matlaba ponavljati mnogo puta. Skript fajlovi ne prihvataju argumente i ne vraćaju vrednosti, i dele adresni prostor sa ostalim skript fajlovima i komandnim interpreterom, drugim rečima, koriste promenljive iz radnog prostora Matlaba.
- Funkcijski fajlovi sadrže funkcije koje mogu da prihvate ulazne argumente, da ih obrade, i generišu izlaz.

M fajlovi

- Komanda *edit ime_fajla* otvara prozor editora m fajlova i priprema *file ime_fajla* za editovanje.
- Skript fajlovi se koriste kad je neophodno neki niz komandi Matlaba ponavljati mnogo puta. Skript fajlovi ne prihvataju argumente i ne vraćaju vrednosti, i dele adresni prostor sa ostalim skript fajlovima i komandnim interpreterom, drugim rečima, koriste promenljive iz radnog prostora Matlaba.
- Funkcijski fajlovi sadrže funkcije koje mogu da prihvate ulazne argumente, da ih obrade, i generišu izlaz.
- Funkcije nemaju pristup radnom prostoru Matlaba, dakle, promenljivama koje radni prostor Matlaba sadrži.

M fajlovi

- Komanda *edit ime_fajla* otvara prozor editora m fajlova i priprema *file ime_fajla* za editovanje.
- Skript fajlovi se koriste kad je neophodno neki niz komandi Matlaba ponavljati mnogo puta. Skript fajlovi ne prihvataju argumente i ne vraćaju vrednosti, i dele adresni prostor sa ostalim skript fajlovima i komandnim interpreterom, drugim rečima, koriste promenljive iz radnog prostora Matlaba.
- Funkcijski fajlovi sadrže funkcije koje mogu da prihvate ulazne argumente, da ih obrade, i generišu izlaz.
- Funkcije nemaju pristup radnom prostoru Matlaba, dakle, promenljivama koje radni prostor Matlaba sadrži.
- Prilikom izvršenja svaka funkcija dobija svoj radni prostor lokalni za funkciju. Promenljive koje sadrži funkcija se nazivaju lokalnim promenljivama funkcije.

Putanje u fajl sistemu

- Komandom *path* dobijamo sadržaj svih direktorijuma koje Matlab pretražuje prilikom pokušaja interpretacije literala.

Putanje u fajl sistemu

- Komandom *path* dobijamo sadržaj svih direktorijuma koje Matlab pretražuje prilikom pokušaja interpretacije literala.
- Komandom *addpath* moguće je dodati izabranu putanju u fajl sistemu u skup putanja koje Matlab pretražuje prilikom interpretacije literala.

Putanje u fajl sistemu

- Komandom *path* dobijamo sadržaj svih direktorijuma koje Matlab pretražuje prilikom pokušaja interpretacije literala.
- Komandom *addpath* moguće je dodati izabranu putanju u fajl sistemu u skup putanja koje Matlab pretražuje prilikom interpretacije literala.

Putanje u fajl sistemu

- Komandom *path* dobijamo sadržaj svih direktorijuma koje Matlab pretražuje prilikom pokušaja interpretacije literala.
- Komandom *addpath* moguće je dodati izabranu putanju u fajl sistemu u skup putanja koje Matlab pretražuje prilikom interpretacije literala. Na primer, naredba $>>$ *addpath*($'e:\direktorijumZaMFajlove'$) dodaće putanju $e:\direktorijumZaMFajlove$ u spisak putanja fajl sistema koje se pretražuju.

Osnovni elementi funkcijskog fajla

```
function [ nFaktorijel ] = faktorijel( n )
% izracunava faktorijel prirodnog broja n
% FAKTORIJEL(N) vraca vrednost faktorijela prirodnog broja N,
% u Matematickoj notaciji, vraca vrednost N!

% Ovo je komentar koji ne pripada help tekstu
nFaktorijel = prod(1:n); end
```

- Linija definicije funkcije *function*[*nFaktorijel*] = *faktorijel*(*n*) definiše funkciju sa imenom *faktorijel*, koja prima jedan ulazni argument (parametar) *n*, i koja vraća vrednost lokalne promenljive *nFaktorijel*.

Osnovni elementi funkcijskog fajla

- Sve linije koda koje počinju znakom % predstavljaju komentare, dakle, linije koda koje se ne interpretiraju.

Osnovni elementi funkcijskog fajla

- Sve linije koda koje počinju znakom % predstavljaju komentare, dakle, linije koda koje se ne interpretiraju.

Osnovni elementi funkcijskog fajla

- Sve linije koda koje počinju znakom % predstavljaju komentare, dakle, linije koda koje se ne interpretiraju. Međutim, prva linija komentara, neposredno ispod linije definicije funkcije, definiše tekst koji dobijamo sa imenom funkcije kad izvršavamo naredbu *help* nad imenom celog direktorijuma, ili kad koristimo naredbu *lookfor*. Ova linija se naziva *H1 linijom*.
- Na primer, ako se m fajl *faktorijel.m* nalazi u direktorijumu $e:\backslash dirZaMF$, dobićemo

```
>> help 'e:\dirZaMF'
Contents of dirZaMF:
faktorijel - izracunava faktorijel prirodnog broja n
```

Osnovni elementi funkcijskog fajla

- Linije *% izracunava faktorijel prirodnog broja n*
  % FAKTORIJEL(N) vraca vrednost faktorijela prirodnog broja N,
  % u Matematickoj notaciji, vraca vrednost N!
  nazivamo *help* tekstom, jer se ovaj tekst dobija prilikom poziva *help faktorijel*. Sledeća prazna linija koda označava prekid *help* teksta, i početak tela funkcije.

Osnovni elementi funkcijskog fajla

- Linije *% izracunava faktorijel prirodnog broja n*
  % FAKTORIJEL(N) vraca vrednost faktorijela prirodnog broja N,
  % u Matematickoj notaciji, vraca vrednost N!
  nazivamo *help* tekstom, jer se ovaj tekst dobija prilikom poziva *help faktorijel*. Sledeća prazna linija koda označava prekid *help* teksta, i početak tela funkcije.
- Unutar tela funkcije mogu se naći linije koje predstavljaju komentar.

Osnovni elementi funkcijskog fajla

- Linije *% izracunava faktorijel prirodnog broja n*
  % FAKTORIJEL(N) vraca vrednost faktorijela prirodnog broja N,
  % u Matematickoj notaciji, vraca vrednost N!
  nazivamo *help* tekstom, jer se ovaj tekst dobija prilikom poziva *help faktorijel*. Sledeća prazna linija koda označava prekid *help* teksta, i početak tela funkcije.
- Unutar tela funkcije mogu se naći linije koje predstavljaju komentar.
- Telo funkcije se završava ključnom rečju *end*. Linija *end* je opciona i može biti izostavljena na kraju tela funkcije.

Osnovni elementi funkcijskog fajla

- Linije *% izracunava faktorijel prirodnog broja n*
  % FAKTORIJEL(N) vraca vrednost faktorijela prirodnog broja N,
  % u Matematickoj notaciji, vraca vrednost N!
  nazivamo *help* tekstom, jer se ovaj tekst dobija prilikom poziva *help faktorijel*. Sledeća prazna linija koda označava prekid *help* teksta, i početak tela funkcije.
- Unutar tela funkcije mogu se naći linije koje predstavljaju komentar.
- Telo funkcije se završava ključnom rečju *end*. Linija *end* je opciona i može biti izostavljena na kraju tela funkcije.
- Sadržaj m fajla, može se prikazati u komandnom prozoru Matlaba komandom *type*.

Osnovni elementi funkcijskog fajla

- Prilikom pisanja m fajlova, kao i prilikom pisanja komandi u komandnom prozoru, linije koda koje se ne završavaju znakom ; proizvode izlaz u komandnom prozoru prilikom interpretiranja.

Osnovni elementi funkcijskog fajla

- Prilikom pisanja m fajlova, kao i prilikom pisanja komandi u komandnom prozoru, linije koda koje se ne završavaju znakom ; proizvode izlaz u komandnom prozoru prilikom interpretiranja.
- Komanda *pause* prilikom interpretiranja prekida izvršenje i čeka se unos sa standardnog ulaza da bi se nastavilo izvršenje.

Osnovni elementi funkcijskog fajla

- Prilikom pisanja m fajlova, kao i prilikom pisanja komandi u komandnom prozoru, linije koda koje se ne završavaju znakom ; proizvode izlaz u komandnom prozoru prilikom interpretiranja.
- Komanda *pause* prilikom interpretiranja prekida izvršenje i čeka se unos sa standardnog ulaza da bi se nastavilo izvršenje.
- Interpretiranje uvek možemo prekinuti pritiskom na kombinaciju *Ctrl-c.*

Imena funkcija i m fajlova

- Bitna stvar prilikom snimanja funkcija u m fajlove je da ime fajla, bez ekstenzije, mora biti isto kao i ime funkcije u fajlu.

Imena funkcija i m fajlova

- Bitna stvar prilikom snimanja funkcija u m fajlove je da ime fajla, bez ekstenzije, mora biti isto kao i ime funkcije u fajlu.
- Imena funkcija se kreiraju na isti način kao imena promenljivih. Da bismo proverili da li neki literal može biti ime funkcije, koristimo naredbu *isvarname*.

Imena funkcija i m fajlova

- Bitna stvar prilikom snimanja funkcija u m fajlove je da ime fajla, bez ekstenzije, mora biti isto kao i ime funkcije u fajlu.
- Imena funkcija se kreiraju na isti način kao imena promenljivih. Da bismo proverili da li neki literal može biti ime funkcije, koristimo naredbu *isvarname*.
- Imena funkcija, takođe, ne mogu biti nizovi karaktera proizvoljne dužine. Naredba *namelengthmax* određuje maksimalnu dozvoljenu dužinu niza karaktera literala koji predstavlja ime funkcije.

Imena funkcija i m fajlova

- Bitna stvar prilikom snimanja funkcija u m fajlove je da ime fajla, bez ekstenzije, mora biti isto kao i ime funkcije u fajlu.
- Imena funkcija se kreiraju na isti način kao imena promenljivih. Da bismo proverili da li neki literal može biti ime funkcije, koristimo naredbu *isvarname*.
- Imena funkcija, takođe, ne mogu biti nizovi karaktera proizvoljne dužine. Naredba *namelengthmax* određuje maksimalnu dozvoljenu dužinu niza karaktera literala koji predstavlja ime funkcije.
- Funkcije, čije ime je isto kao i ime m fajla u kome su definisane nazivaju se primarne funkcije.

Naredba if

```
if izraz01
blok_naredbi01
else
blok_naredbi02
end
```

- Izraz *izraz01* ima vrednost *true*, ako predstavlja neprazan višedimenzionalni niz čiji su svi elementi različiti od nule (elementi niza su tipa *logical* ili *numeric*), u ostalim slučajevima vrednost izraza *izraz01* ima vrednost *false.*

Naredba if

```
if izraz01
blok_naredbi01
else
blok_naredbi02
end
```

- Izraz *izraz01* ima vrednost *true*, ako predstavlja neprazan višedimenzionalni niz čiji su svi elementi različiti od nule (elementi niza su tipa *logical* ili *numeric*), u ostalim slučajevima vrednost izraza *izraz01* ima vrednost *false.*
- Napisati funkciju *fun01* koja izračunava apsolutnu vrednost argumenta.

Naredba if

```
function y = fun01( x )
if x > 0
y = x;
else
y = -x;
end
end
```

- >> fun01(−1), >> fun01([1 2 3]), >> fun01([1 0 3]).

Naredba if

```
function y = fun01( x )
if x > 0
y = x;
else
y = -x;
end
end
```

- >> *fun*01(−1), >> *fun*01([1 2 3]), >> *fun*01([1 0 3]).
- U trećem primeru argument funkcije *fun01* ima jedan element jednak nuli, pa uslov *if* naredbe ima vrednost *false.*

Naredba if

- Napisati funkciju *fun02* koja ima vrednost jedan u tački nula, i vrednost nula u ostalim tačkama.

```
 if izraz01
blok_naredbi01
elseif izraz02
blok_naredbi02
else
blok_naredbi03
end
```

Naredba return

- Ako želimo da napustimo izvršenje funkcije, pre nego što smo došli do kraja tela funkcije, koristimo naredbu *return.*

Naredba return

- Ako želimo da napustimo izvršenje funkcije, pre nego što smo došli do kraja tela funkcije, koristimo naredbu *return.*
- Funkciju koja izračunava skalarni proizvod dva vektora (ako je pozovemo sa argumentima koji nisu vektori kolone treba napustiti izvršenje).

Naredba return

- Ako želimo da napustimo izvršenje funkcije, pre nego što smo došli do kraja tela funkcije, koristimo naredbu *return.*
- Funkciju koja izračunava skalarni proizvod dva vektora (ako je pozovemo sa argumentima koji nisu vektori kolone treba napustiti izvršenje).

```
function z = skalPro( x, y )
sizeX = size(x);
sizeY = size(y);
ifsizeX(2) ~= 1 || sizeY(2) ~= 1 || sizeX(1) ~= sizeY(1)
disp('uneseni vektori nisu vektori kolone istog tipa');
return;
end
z = y' * x; end
```

Naredba switch

```
switch izraz
    case izraz01     blok_naredbi01
    case izraz02     blok_naredbi02
    :
    otherwise        blok_naredbi
end
```

- Matlab ne zahteva upotrebu naredbe *break* da bi se izbeglo dalje pretraživanje vrednosti izraza.

Naredba switch

```
switch izraz
    case izraz01      blok_naredbi01
    case izraz02      blok_naredbi02
    :
    otherwise         blok_naredbi
end
```

- Matlab ne zahteva upotrebu naredbe *break* da bi se izbeglo dalje pretraživanje vrednosti izraza.
- Izraz *izraz* može biti skalarna vrednost numeričkog tipa ili string. U slučaju da je vrednost izraza *izraz* tipa string, jednakost se ispituje funkcijom *strcmp.*

Naredba switch

- Napisati funkciju koja zaključuje da li je dan u nedelji radni ili neradni.

```
function y = fun08(x)
switch lower(deblank(x))
        case 'subota', 'nedelja'
            y = 'neradni dan';
        case 'ponedeljak', 'utorak', 'sreda', 'cetvrtak', 'petak'
            y = 'radni dan';
        otherwise
            y = 'nije dan u nedelji';
end
```

Naredba for

```
for indeks = vrednosti
        blok_naredbi
end
```

Naredba for

```
for indeks = vrednosti
        blok_naredbi
end
```

- Napisati funkciju koja izračunava faktorijel prirodnog broja.

Naredba for

```
for indeks = vrednosti
        blok_naredbi
end
```

- Napisati funkciju koja izračunava faktorijel prirodnog broja.

```
function y = fun01( x )
%FUN01(X) izracunava faktorijel unetog broja X
% da bi funkcija ispravno radila uneta vrednost
% mora biti matrica tipa 1*1
y = 1;
for i = 1:x
    y = y*i;
end
end
```

Naredba for

```
function y = fun02( x )
%FUN02(X) izracunava proizvod elemenata vektora X tipa 1*n
y = 1;
for i = x
y = y*i;
end
end
```

Naredba for

```
function y = fun02( x )
%FUN02(X) izracunava proizvod elemenata vektora X tipa 1*n
y = 1;
for i = x
y = y*i;
end
end
```

```
>> fun02([5 2 10])
```

```
>> fun02([5; 2; 10])
```

Naredba for

```
function y = fun02( x )
%FUN02(X) izracunava proizvod elemenata vektora X tipa 1*n
y = 1;
for i = x
y = y*i;
end
end
```

```
>> fun02([5 2 10])        >> fun02([5; 2; 10])
```

U prvom pozivu indeks *for* petlje, promenljiva *i*, uzima vrednosti iz vektora [5 2 10] i njima množi vrednost izlaznog argumenta *y*.

Naredba for

```
function y = fun03( x )
%FUN03(X) formira matricu sa obrnutim rasporedom kolona u odnosu na X
y = [];
for i = x
y = [i y];
end
end
```

Naredba for

```
function y = fun03( x )
%FUN03(X) formira matricu sa obrnutim rasporedom kolona u
odnosu na X
y = [];
for i = x
y = [i y];
end
end
```

```
>> fun03([1 2 3; 4 5 6])
```

Naredba for

```
function y = fun03( x )
%FUN03(X) formira matricu sa obrnutim rasporedom kolona u
odnosu na X
y = [];
for i = x
y = [i y];
end
end
```

```
>> fun03([1 2 3; 4 5 6])
```

Indeks *for* petlje nije lokalna promenljiva *for* petlje.

Petlja while

while izraz
blok_naredbi
end

Petlja while

```
while izraz
        blok_naredbi
end
```

- Vrednost izraza *izraz* je *true* ako je izraz neprazan višedimenzionalan niz i ako ne sadrži nijedan element koji ima vrednost nula ili *false*, u ostalim slučajevima vrednost izraza je *false*.
- Primer pronalaženja najmanjeg prirodnog broja čija je vrednost faktorijela veća od neke unapred zadate vrednosti.

Petlja while

```
function y = fun04( x )
%FUN04(X) pronalazi najmanji prirodan broj y takav da je
y! >= X
y = 1;
fakt = 1;
while fakt < x
        y = y+1;
        fakt = fakt * y;
end
end
```

Petlja while

```
function y = fun04( x )
%FUN04(X) pronalazi najmanji prirodan broj y takav da je
y! >= X
y = 1;
fakt = 1;
while fakt < x
        y = y+1;
        fakt = fakt * y;
end
end
```

```
>> fun04(100)        >> fun04([100 200 300])
```

Naredbe continue i break

- Naredba break se korisiti kada želimo da prekinemo izvršenje petlje.

Naredbe continue i break

- Naredba break se korisiti kada želimo da prekinemo izvršenje petlje.
- Naredba continue služi da se pređe na sledeći ciklus u petlji bez završetka prethodnog.

Naredbe continue i break

- Naredba break se korisiti kada želimo da prekinemo izvršenje petlje.
- Naredba continue služi da se pređe na sledeći ciklus u petlji bez završetka prethodnog.
- Napisati funkcija koja koristeći naredbu *continue* izračunava zbir prvih x neparnih brojeva.

Naredbe continue i break

```
function y = fun06( x )
%FUN06(X) izracunava zbir prvih x prirodnih brojeva
y = 0;
i = 0;
while i ¡ x
        i = i+1;
        if mod(i,2) == 0
        continue;
        end
        y = y+i;
end
end
```

Funkcija error

- Funkcija *error* služi za prekidanje izvršenja funkcije uz štampanje poruke na standardnom izlazu.

Funkcija error

- Funkcija *error* služi za prekidanje izvršenja funkcije uz štampanje poruke na standardnom izlazu.
- *error(error_poruka)* prekida interpretaciju funkcije i štampa na standardni izlaz argument *error_poruka*, koji je tipa *string.*

Funkcija error

- Funkcija *error* služi za prekidanje izvršenja funkcije uz štampanje poruke na standardnom izlazu.
- *error(error_poruka)* prekida interpretaciju funkcije i štampa na standardni izlaz argument *error_poruka*, koji je tipa *string.*

```
function [y]=log01(x)
%LOG01(X) racuna logaritam pozitivnog broja X
if ~(x > 0)
error('argument funkcije mora biti pozitivan');
end
y=log(x);
end
```

Funkcija find

- Funkcija *find* omogućava efikasno pronalaženje indeksa elemenata matrice koji zadovoljavaju izvestan uslov.

Funkcija find

- Funkcija *find* omogućava efikasno pronalaženje indeksa elemenata matrice koji zadovoljavaju izvestan uslov.
- Ako je funkcija *find* pozvana sa jednim izlaznim argumentnom, funkcija vraca linearne indekse elemenata koji zadovoljavaju uslov. Ako je funkcija pozvana sa dva izlazna argumenta funkcija *find* vraća vektore, pri čemu je prvi vektor vektor indeksa vrste, a drugi vektor je vektor indeksa kolona elemenata koji zadovoljavaju uslov funkcije *find*.

Funkcija find

- Funkcija *find* omogućava efikasno pronalaženje indeksa elemenata matrice koji zadovoljavaju izvestan uslov.
- Ako je funkcija *find* pozvana sa jednim izlaznim argumentnom, funkcija vraca linearne indekse elemenata koji zadovoljavaju uslov. Ako je funkcija pozvana sa dva izlazna argumenta funkcija *find* vraća vektore, pri čemu je prvi vektor vektor indeksa vrste, a drugi vektor je vektor indeksa kolona elemenata koji zadovoljavaju uslov funkcije *find*.
- Na primer, ako želimo da pronađemo indekse elemenata matrice koji imaju vrednost veću od 0, možemo koristiti sledeće naredbe

```
>> A = [-1  -10 2; 5  -3 6];
>> a = find(A > 0)        >> [n, m] = find(A > 0)
```

Ulazni i izlazni argumenti funkcije

- Da bi se omogućilo da funkcije mogu imati više izlaznih argumenata, svi izlazni argumenti se grupišu u matricu tipa $1 \times n$.

Ulazni i izlazni argumenti funkcije

- Da bi se omogućilo da funkcije mogu imati više izlaznih argumenata, svi izlazni argumenti se grupišu u matricu tipa $1 \times n$.
- Funkcija *fun1* koja vraća argumente *a*, *b* i *c*, a koja nema ulazne argumente, može imati neku od sledećih definicija

Ulazni i izlazni argumenti funkcije

- Da bi se omogućilo da funkcije mogu imati više izlaznih argumenata, svi izlazni argumenti se grupišu u matricu tipa $1 \times n$.
- Funkcija *fun1* koja vraća argumente *a*, *b* i *c*, a koja nema ulazne argumente, može imati neku od sledećih definicija

$$function[a, b, c] = fun1$$
$$function[a, b, c] = fun1()$$

Ulazni i izlazni argumenti funkcije

- Da bi se omogućilo da funkcije mogu imati više izlaznih argumenata, svi izlazni argumenti se grupišu u matricu tipa $1 \times n$.
- Funkcija *fun1* koja vraća argumente *a*, *b* i *c*, a koja nema ulazne argumente, može imati neku od sledećih definicija

$$function[a, b, c] = fun1$$
$$function[a, b, c] = fun1()$$

- Funkcija *fun2* nema izlazne argumente, a nema ni ulazne.

Ulazni i izlazni argumenti funkcije

- Da bi se omogućilo da funkcije mogu imati više izlaznih argumenata, svi izlazni argumenti se grupišu u matricu tipa $1 \times n$.
- Funkcija *fun1* koja vraća argumente *a*, *b* i *c*, a koja nema ulazne argumente, može imati neku od sledećih definicija

$$function[a, b, c] = fun1$$
$$function[a, b, c] = fun1()$$

- Funkcija *fun2* nema izlazne argumente, a nema ni ulazne.

$$functionfun2$$
$$function[] = fun2;$$
$$function[] = fun2(); \qquad functionfun2();$$

Ulazni i izlazni argumenti funkcije

- Funkcije *nargin* i *nargout*, koriste se za određivanje broja ulaznih i izlaznih argumenata funkcije.

Ulazni i izlazni argumenti funkcije

- Funkcije *nargin* i *nargout*, koriste se za određivanje broja ulaznih i izlaznih argumenata funkcije.

```
>> a = [1 2 3; 4 5 6];
>> size(a),                    >> [n, m] = size(a)
```

Ulazni i izlazni argumenti funkcije

- Funkcije *nargin* i *nargout*, koriste se za određivanje broja ulaznih i izlaznih argumenata funkcije.

```
>> a = [1 2 3; 4 5 6];
>> size(a),                    >> [n, m] = size(a)
```

```
function[a, b] = fun3(x, y, z, u)
a = 1;
b = 2;
end
```

Ulazni i izlazni argumenti funkcije

- Funkcije *nargin* i *nargout*, koriste se za određivanje broja ulaznih i izlaznih argumenata funkcije.

```
>> a = [1 2 3; 4 5 6];
>> size(a),                    >> [n, m] = size(a)
```

```
function[a, b] = fun3(x, y, z, u)
a = 1;
b = 2;
end
```

```
>> n = fun3(1, 2, 3, 4)        >> [n, m] = fun3(1, 2, 3, 4)
```

Ulazni i izlazni argumenti funkcije

- *nargin*, *nargout*, *narginchk*, *nargoutchk* omogućavaju različito ponašanje funkcije u zavisnosti od broja ulaznih i izlaznih argumenata.

Ulazni i izlazni argumenti funkcije

- *nargin, nargout, narginchk, nargoutchk* omogućavaju različito ponašanje funkcije u zavisnosti od broja ulaznih i izlaznih argumenata.

```
function [x y] = pak(u, v)
%PAK(U,V) pakuje ulazne podatke u izlazne narginchk(1,2),
nargoutchk(0,2);
switch nargin
        case 1
            switch nargout
                case {0, 1}          x = u;
                case 2                x = u;
```

Ulazni i izlazni argumenti funkcije

```
                    y = u;
            otherwise       %nemoguce zbog nargoutchk
end
    case 2
        switch nargout
            case 0, 1       x = [u v];
            case 2          x = u; y = v;
            otherwise       %nemoguce zbog nargoutchk
end
otherwise    %nemoguce zbog narginchk
end
end
```

Ulazni i izlazni argumenti funkcije

- Funkcija *pak* iako definisana sa dva ulazna parametra ne proizvodi grešku kad je pozvana sa jednim argumentom.

Ulazni i izlazni argumenti funkcije

- Funkcija *pak* iako definisana sa dva ulazna parametra ne proizvodi grešku kad je pozvana sa jednim argumentom.

```
>> x = pak(1)
>> [x, y] = pak(1)
>> x = pak(1, 2)
>> [x, y] = pak(1, 2)
>> [x, y] = pak
```

- Funkcije *narginchk* i *nargoutchk* prihvataju dva argumenta koji određuju minimalni i maksimalni broj ulaznih i izlaznih argumenata redom. U slučaju da je broj ulaznih ili izlaznih argumenata van ovih granica, dobijamo poruku o grešci.

Globalne promenljive

- Globalne promenljive su deljive među svim radnim prostorima.

Globalne promenljive

- Globalne promenljive su deljive među svim radnim prostorima.
- Da bi neki radni prostor imao pristup nekoj globalnoj promenljivoj, promenljiva, u tom radnom prostoru, mora biti deklarisana kao globalna.

Globalne promenljive

- Globalne promenljive su deljive među svim radnim prostorima.
- Da bi neki radni prostor imao pristup nekoj globalnoj promenljivoj, promenljiva, u tom radnom prostoru, mora biti deklarisana kao globalna.
- Promenljiva se deklariše kao globalna upotrebom ključne reči *global*.

Globalne promenljive

- Globalne promenljive su deljive među svim radnim prostorima.
- Da bi neki radni prostor imao pristup nekoj globalnoj promenljivoj, promenljiva, u tom radnom prostoru, mora biti deklarisana kao globalna.
- Promenljiva se deklariše kao globalna upotrebom ključne reči *global*.

```
function [x]=funGlobal
%FUNGLOBAL sluzi za demonstraciju globalne promenljive
global xGlobal;
x=xGlobal;
end
```

Globalne promenljive

```
>> funGlobal
>> xGlobal = 3;          >> funGlobal
```

Globalne promenljive

```
>> funGlobal
>> xGlobal = 3;          >> funGlobal
```

Da bi radni prostor Matlaba dobio pristup globalnoj promenljivoj *xGlobal*, moramo ovu promenljivu deklarisati kao globalnu unutar radnog prostora Matlaba.

Globalne promenljive

```
>> funGlobal
>> xGlobal = 3;          >> funGlobal
```

Da bi radni prostor Matlaba dobio pristup globalnoj promenljivoj *xGlobal*, moramo ovu promenljivu deklarisati kao globalnu unutar radnog prostora Matlaba.

```
>> clear xGlobal;
>> global xGlobal;          >> xGlobal = 3;
>> funGlobal
```

Globalne promenljive

```
>> funGlobal
>> xGlobal = 3;          >> funGlobal
```

Da bi radni prostor Matlaba dobio pristup globalnoj promenljivoj *xGlobal*, moramo ovu promenljivu deklarisati kao globalnu unutar radnog prostora Matlaba.

```
>> clear xGlobal;
>> global xGlobal;          >> xGlobal = 3;
>> funGlobal
```

Promenljivu smo prvo morali da obrišemo, pre nego što smo je deklarisali kao globalnu, jer smo je, u prethodnom nizu naredbi, deklarisali kao lokalnu.

Statične promenljive

- Statične promenljive čuvaju svoju vrednost i po završetku interpretiranja funkcije u kojoj su deklarisane i dostupne su samo u radnom prostoru funkcije u kojoj su deklarisane.

Statične promenljive

- Statične promenljive čuvaju svoju vrednost i po završetku interpretiranja funkcije u kojoj su deklarisane i dostupne su samo u radnom prostoru funkcije u kojoj su deklarisane.
- Statične promenljive se deklarišu upotrebom ključne reči *persistent*.

Statične promenljive

- Statične promenljive čuvaju svoju vrednost i po završetku interpretiranja funkcije u kojoj su deklarisane i dostupne su samo u radnom prostoru funkcije u kojoj su deklarisane.
- Statične promenljive se deklarišu upotrebom ključne reči *persistent*.

```
function [x]=funPersistent
%FUNPERSISTENT sluzi za ilustraciju staticke promenljive
persistent xPersistent;
if isempty(xPersistent)
xPersitent=0;
end
x=xPersistent;
xPersistent=xPersistent+1;
end
```

Statične promenljive

- Pozovimo funkciju *funPersitent* nekoliko puta.

Statične promenljive

- Pozovimo funkciju *funPersitent* nekoliko puta.
- Ovaj koncept je osmišljen za one situacije kad je potrebno preneti rezultate obrade između dva poziva funkcije.

Statične promenljive

- Pozovimo funkciju *funPersitent* nekoliko puta.
- Ovaj koncept je osmišljen za one situacije kad je potrebno preneti rezultate obrade između dva poziva funkcije.
- Statične promenljive nestaju kad se prekine proces Matlaba.

Statične promenljive

- Pozovimo funkciju *funPersitent* nekoliko puta.
- Ovaj koncept je osmišljen za one situacije kad je potrebno preneti rezultate obrade između dva poziva funkcije.
- Statične promenljive nestaju kad se prekine proces Matlaba.
- Moguće je izazvati nestanak statičnih promenljivih i pozivanjem funkcije *clear* sa imenom funkcije, ili pozivom *clear all.*

Podfunkcije

- Funkcije koje su definisane u m fajlovima, a čije ime nije isto sa imenom fajla nazivaju se podfunkcije funkcije i ne mogu se pozvati na izvršenje iz komandnog prozora.

Podfunkcije

- Funkcije koje su definisane u m fajlovima, a čije ime nije isto sa imenom fajla nazivaju se podfunkcije funkcije i ne mogu se pozvati na izvršenje iz komandnog prozora.
- Ove funkcije se mogu pozvati iz ostalih podfunkcija koje se nalaze u okviru istog m fajla ili iz primarne funkcije u okviru tog m fajla, dakle, funkcije čije je ime isto kao i ime fajla.

Podfunkcije

- Funkcije koje su definisane u m fajlovima, a čije ime nije isto sa imenom fajla nazivaju se podfunkcije funkcije i ne mogu se pozvati na izvršenje iz komandnog prozora.
- Ove funkcije se mogu pozvati iz ostalih podfunkcija koje se nalaze u okviru istog m fajla ili iz primarne funkcije u okviru tog m fajla, dakle, funkcije čije je ime isto kao i ime fajla.
- Definicija primarne funkcije mora biti prva, a iza njene definicije dolaze definicije ostalih podfunkcija.

Podfunkcije

- Funkcije koje su definisane u m fajlovima, a čije ime nije isto sa imenom fajla nazivaju se podfunkcije funkcije i ne mogu se pozvati na izvršenje iz komandnog prozora.
- Ove funkcije se mogu pozvati iz ostalih podfunkcija koje se nalaze u okviru istog m fajla ili iz primarne funkcije u okviru tog m fajla, dakle, funkcije čije je ime isto kao i ime fajla.
- Definicija primarne funkcije mora biti prva, a iza njene definicije dolaze definicije ostalih podfunkcija.
- Svaka od podfunkcija ima sopstveni radni prostor, dakle, ne može se pristupati lokalnim promenljivama jedne podfunkcije iz druge podfunkcije.

Podfunkcije

```
function [ nFaktorijel ] = faktorijel(n)
% izracunava faktorijel prirodnog broja n
% FAKTORIJEL(N) vraca vrednost faktorijela prirodnog broja N,
% u Matematickoj notaciji, vraca vrednost N!
nFaktorijel = proizvod(n);
function b = proizvod(n)
% ovo je podfunkcija koja racuna proizvod elemenata vektora
b = prod(lista(n));
function c = lista(n)
% ova podfunkcija kreira listu prvih n prirodnih brojeva
c = 1:n;
```

Anonimne funkcije i pokazivači na funkcije

- Definicija funkcije može biti data i koristeći jednu jedinu liniju koda, bez snimanja u m fajl.

Anonimne funkcije i pokazivači na funkcije

- Definicija funkcije može biti data i koristeći jednu jedinu liniju koda, bez snimanja u m fajl.

```
>> kvadrat = @(x)x.^2;
>> kvadrat([3 4 5])
```

Anonimne funkcije i pokazivači na funkcije

- Definicija funkcije može biti data i koristeći jednu jedinu liniju koda, bez snimanja u m fajl.

```
>> kvadrat = @(x)x.^2;
>> kvadrat([3 4 5])
```

- Generički način na koji se definiše anonimna funkcija:

Anonimne funkcije i pokazivači na funkcije

- Definicija funkcije može biti data i koristeći jednu jedinu liniju koda, bez snimanja u m fajl.

```
>> kvadrat = @(x)x.^2;
>> kvadrat([3 4 5])
```

- Generički način na koji se definiše anonimna funkcija:

$$f = @(listaArgumenata)\ teloFunkcije$$

Anonimne funkcije i pokazivači na funkcije

- Definicija funkcije može biti data i koristeći jednu jedinu liniju koda, bez snimanja u m fajl.

```
>> kvadrat = @(x)x.^2;
>> kvadrat([3 4 5])
```

- Generički način na koji se definiše anonimna funkcija:

$$f = @(listaArgumenata)\ teloFunkcije$$

Restartovanjem Matlaba definicije anonimnih funkcija nestaju.

Anonimne funkcije i pokazivači na funkcije

- Za svaku definisanu funkciju ili korisnički definisanu funkciju moguće je dobiti promenljivu koja predstavlja tu funkciju a koja je tipa *function_handle*.

Anonimne funkcije i pokazivači na funkcije

- Za svaku definisanu funkciju ili korisnički definisanu funkciju moguće je dobiti promenljivu koja predstavlja tu funkciju a koja je tipa *function_handle*.
- Podaci tipa *function_handle* su ekvivalent tipa pokazivača na funkcije u programskom jeziku C.

Anonimne funkcije i pokazivači na funkcije

- Za svaku definisanu funkciju ili korisnički definisanu funkciju moguće je dobiti promenljivu koja predstavlja tu funkciju a koja je tipa *function_handle*.
- Podaci tipa *function_handle* su ekvivalent tipa pokazivača na funkcije u programskom jeziku C.
- Funkcija *feval* prima kao prvi argument pokazivač na funkciju, a kao ostale argumente argumente na koje će pokazivač biti primenjen.

Anonimne funkcije i pokazivači na funkcije

- Za svaku definisanu funkciju ili korisnički definisanu funkciju moguće je dobiti promenljivu koja predstavlja tu funkciju a koja je tipa *function_handle*.
- Podaci tipa *function_handle* su ekvivalent tipa pokazivača na funkcije u programskom jeziku C.
- Funkcija *feval* prima kao prvi argument pokazivač na funkciju, a kao ostale argumente argumente na koje će pokazivač biti primenjen.

```
>> sin_handle = @sin;
>> [sin(4) sin_handle(4)]
>> fSin = @sin;          >> feval(fSin, 1)
```

Ugnježdjene funkcije

- Ugnježdena funkcija je definisana i implementirana unutar tela primarne funkcije ili podfunkcije.

Ugnježdjene funkcije

- Ugnježdena funkcija je definisana i implementirana unutar tela primarne funkcije ili podfunkcije.
- Ugnježdena funkcija se može koristiti samo unutar funkcije u kojoj je definisana i ima pristup radnom prostoru, dakle, promenljivama, funkcije u koju je ugnježdena.

```
function y = faktorijel( x )
%FAKTORIJEL(X) racuna faktorijel od X
y = prod(niz);
function z = niz
z = 1:x;
end
end
```

Ugnježdjene funkcije

- Funkcija *faktorijel* ne može koristiti vrednosti promenljivih iz radnog prostora funkcije *niz*, jer radni prostor funkcije *niz* nestaje kad se dođe do kraja funkcije *niz*.

Ugnježdjene funkcije

- Funkcija *faktorijel* ne može koristiti vrednosti promenljivih iz radnog prostora funkcije *niz*, jer radni prostor funkcije *niz* nestaje kad se dođe do kraja funkcije *niz*.
- Ako ugnježdena funkcija sadrži lokalnu promenljivu sa istim imenom, recimo x, kao i funkcija u koju je ugnježdena, onda interpreter neće unutar radnog prostora ugnježdene funkcije kreirati novu lokalnu promenljivu x.

Ugnježdjene funkcije

```
function fun
x = 1;
disp('x u f pre g'), disp(x);
g;
function g
x = 2;
disp('x u g'), disp(x);
end
disp('x u f posle g'), disp(x);
end
```

Skript fajlovi

- Skript fajlovi nemaju posebni radni prostor.

Skript fajlovi

- Skript fajlovi nemaju posebni radni prostor.
- Unos podataka prilikom izvršenja skript fajlova, ili prilikom izvršenja funkcija, funkcijom *input*.

Skript fajlovi

- Skript fajlovi nemaju posebni radni prostor.
- Unos podataka prilikom izvršenja skript fajlova, ili prilikom izvršenja funkcija, funkcijom *input*.

```
% ovo je skript fajl koji racuna
% srednju vrednost matrice A
[n, m] = size(A);
srA = sum(sum(A))/(n * m);
```

Skript fajlovi

- Skript fajlovi nemaju posebni radni prostor.
- Unos podataka prilikom izvršenja skript fajlova, ili prilikom izvršenja funkcija, funkcijom *input*.

```
% ovo je skript fajl koji racuna
% srednju vrednost matrice A
[n, m] = size(A);
srA = sum(sum(A))/(n * m);
```

```
input('unesite matricu:');
disp(A');
```