

Funkcije ispisa i upisa

Funkcija `input`

Funkcija `input` omogućuje unos podataka tijekom izvođenja programa napisanog u skript datoteci. Sastavljena je od ključne riječi `input` i imena varijable kojoj će se pridružiti unesena vrijednost.

Sintaksa naredbe `input`:

```
varijabla = input('tekst koji će biti prikazan u naredbenom prozoru');
```

Nakon pokretanja skript datoteke i izvođenja programske linije s funkcijom `input`, u naredbenom prozoru se prikazuje poruka nakon koje se pojavljuje treptajući kursor. To je poruka korisniku da putem tipkovnice utipka vrijednost koja će biti pridružena varijabli. Nakon upisivanja vrijednosti potrebno je pritisnuti tipku Enter. Time će varijabli biti pridružena upisana vrijednost.

Funkcija `disp`

Funkcija `disp` ispisuje vrijednost varijable bez ispisa njezina imena.

Sintaksa:

```
disp (ime_varijable)
```

Funkcija `fprintf`

Pomoću funkcije `fprintf` možemo ispisivati rezultate (tekst i podatke) na ekranu ili ih zapisivati u datoteku.

Prilikom ispisa moguće je formatirati rezultat, npr. tekst i numeričke vrijednosti mogu biti prikazani u istom retku, a moguće je zadati i format brojeva.

Za zajednički prikaz teksta i broja (vrijednosti varijable) rabimo funkciju `fprintf` čija je sintaksa:

```
fprintf('tekst kao niz znakova %5.2f dodatni tekst',ime_varijable)
```

Znak `%` označava mjesto gdje broj treba umetnuti u tekst.

ime_varijable označava varijablu čija se vrijednost prikazuje.

Oznaka `%-5.2f` definira format broja: **f** je znak konverzije (obvezno - tj. zapis realnog broja). **5** ovdje označava ukupnu širinu polja, **.2** označava preciznost (neobvezno) na 2 decimale.

Primjer: Računanje prosječne vrijednosti koncentracije u dva mjerenja

```
% Unos podataka za mjerenje koncentracije  
concl = input('Unesi prvo mjerenje koncentracije: ');
```

```

conc2 = input('Unesi drugo mjerenje koncentracije: ');
% Računanje prosječne vrijednosti koncentracije
prosje_koncentracije = (conc1 + conc2) / 2;
% Ispis rezultata
fprintf('Prosje_koncentracije je: %5.2f\n', prosje_koncentracije);

```

Zadatak za vježbu: *Napišite program za računanje prosječnog broja bodova postignutih na dva testa. Rezultat prikazite na jednu decimalu.*

Naredbe odluke i ponavljanja

Grananje - struktura `if end`

Izraz u `if` naredbi odluke može biti sastavljen od **relacijskih** (<, <=, >, >=, ==, ~=) i/ili **logičkih** operatora (&, |, ~). Ako je logički izraz u iskazu `if` *istinit* program izvodi niz naredbi ili funkcija između `if` i `end`. Ako je logički izraz *neistinit*, odnosno, uvjet nije ispunjen, izlazi se iz tog dijela programa, tj. program preskače naredbe između iskaza `if` i `end` i nastavlja izvršavati naredbe iza iskaza `end`.

```

if logički_izraz
naredbe ili funkcije;
end

```

Primjer:

```

prvi_broj= 2;
drugi_broj = 3;

if prvi_broj > drugi_broj
disp('prvi je veci')
end

if prvi_broj < drugi_broj
disp('drugi broj je veci')
end

```

Grananje - struktura `if else end`

```

if izraz
naredbe ili funkcije
else
naredbe ili funkcije
end

```

Ako je izraz (uvjet) u `if` ispunjen, izvršavaju se naredbe ili funkcije između `if` i `else`. Ako uvjet nije ispunjen, izvršavaju se naredbe ili funkcije između `else` i `end`.

Primjer: Varijabli **x** slučajno se dodjeljuje realna vrijednost u rasponu [0,1]. Zatim se ispituje je li vrijednost varijable **x** manja od 0,5. Ukoliko je vrijednost varijable **x** < 0,5 ispisuje se "pismo", a ukoliko je veća od ili jednaka 0,5, ispisuje se "glava".

```
x = rand
if x < 0.5
disp('pismo')
else
disp('glava')
end
```

Zadatak za vježbu: Unesite dva proizvoljna broja a i b. Ako vrijedi $a > b$ i $b > 5$, izračunajte njihov zbroj. U suprotnom izračunajte njihovu razliku.

Cikličke strukture

Niz algoritamskih koraka u kojem se jedan ili više koraka može izvršiti više puta pri jednom izvršavanju algoritma zadatka tvori **cikličku algoritamsku shemu**.

Ciklička struktura realizira se u obliku programske **petlje** (engl. *loop*). Svako izvršavanje petlje zove se **prolaz** (engl. *pass*) ili iteracija. U svakom prolazu se barem jednoj varijabli definiranoj unutar petlje pridružuju nove vrijednosti.

U naredbe ponavljanja ubrajaju se **for** petlja i **while** petlja. Kod **for** petlje najčešće se dio programa ponavlja unaprijed određeni broj puta dok se kod **while** petlje najčešće dio programa ponavlja dok se ne zadovolji uvjet. Broj ponavljanja dijela programa unutar **while** petlje nije unaprijed poznat kao što je to kod **for** petlje.

Izvršavanje obje vrste petlji može se prekinuti u svakom trenutku naredbom **break**.

for-end petlja

for petlje omogućuju ponavljanje izvršavanja dijela koda. Obično se koriste za specificiranje unaprijed definiranog broja iteracija (broja ponavljanja odnosno broj prolaza kroz petlju).

U petljama tipa **for-end** izvršavanje naredbi ponavlja se zadani broj puta.

```
for k = f:s:t
...niz naredbi
end
```

k je kontrolna varijabla petlje, **brojač** petlje. Taj dio određuje koliko će se puta dio programa između **for** i **end** dijela izvršavati.

f je vrijednost **k** u prvom prolazu

s je **korak** povećanja **k** nakon svakog prolaza

t je vrijednost **k** u zadnjem prolazu

Petlja mijenja vrijednost brojača od vrijednosti **f** do vrijednosti **t** uz povećavanje vrijednosti određene vrijednošću **s**.

- U prvom prolazu **k=f** računalo izvodi naredbe između naredbi **for** i **end**. Zatim se program vraća na naredbu **for** da bi započeo sljedeći prolaz. Varijabla **k** dobiva novu vrijednost, jednaku **k = f + s**, a zatim se s tom novom vrijednošću **k** izvode naredbe između **for** i **end**. Postupak se ponavlja dok u posljednjem koraku **k** ne postane jednako **t**. U tom slučaju program se ne vraća na **for**, već nastavlja s naredbama iza naredbe **end**.
- Npr., ako je **k = 1:2:9**, imamo pet prolaza kroz petlju, a vrijednost **k** u svakom prolazu kroz petlju je 1, 3, 5, 7 i 9.
- Korak petlje može biti i negativan (npr. **k = 30:-5:15** daje četiri prolaza u kojima je **k = 30, 25, 20, 15**).
- Ako nije zadana vrijednost koraka petlje, **s**, podrazumijeva se 1.
- Svako naredbi **for** u programu mora biti pridružena i odgovarajuća naredba **end**!
- Kada se petlja završi, kontrolna varijabla (**k**) ima vrijednost koja joj je bila posljednja dodijeljena.

Primjer: Izračunajte kvadrate brojeva od 1 do 5 koristeći **for** petlju.

```
for x = 1:5
y = x^2; % kontrolna varijabla se može koristiti za računanje
fprintf('%g\n', y)
end
```

Primjer: Izračunajte zbroj brojeva od 1 do 50 koristeći **for** petlju.

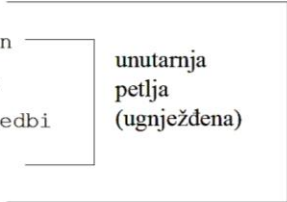
Petlja se ponavlja 50 puta i pri svakom prolasku kroz petlju varijabli **x** dodaje se vlastita vrijednost iz prijašnjeg koraka i vrijednost brojača petlje **j**.

Na početku se varijabli **x** dodjeljuje vrijednost 0 kako bi zbroj na početku programa bio jednak nuli.

```
x = 0;
for j = 1:50
x = x + j;
end
x
```

Mogu se rabiti i tzv. **ugniježdene for** petlje čija je sintaksa sljedeća:

```
for i = 1:m
    for j = 1:n
        ... niz
        ... naredbi
    end
end
```



Ugniježdene **for** petlja se sastoji od dvije **for** petlje. To znači da petlja može započeti (i završiti

se) unutar druge petlje.

Primjer: Program u matricu x dimenzija 10×10 upisuje vrijednosti tablice množenja brojeva od 1 do 10.

Vanjska petlja određuje red matrice, a unutarnja petlja stupac matrice. U svaki element matrice upisuje se umnožak brojača petlji m i n . Vanjska **for** petlje ponavlja se 10 puta, a unutar svakog prolaza izvršava se unutarnja **for** petlja koja se također ponavlja 10 puta. Zbog toga se naredba unutar unutarnje petlje ukupno izvrši 100 puta.

```
for m = 1:10
for n = 1:10
x(m,n) = m * n;
end
end
x
```

while-end petlja

while petlja omogućuje ponavljanje izvršavanja koda na temelju zadanog logičkog uvjeta. Ovakav oblik petlje se rabi kad nije unaprijed poznat broj ponavljanja petlje (broj prolaza), a petlja se izvršava dok ne bude ispunjen zadani uvjet.

Sintaksa **while** petlje ima sljedeći oblik:

```
while uvjet
naredba ili funkcija
...
end
```

Dio programa između **while** i **end** izvršavat će se sve dok je ispunjen uvjet. Kada uvjet više nije ispunjen, izvršavanje petlje se prekida.

Primjer: Izračunajte kvadrate brojeva od 1 do 5 koristeći **while** petlju.

```
x = 1;
while x<=5
y = x^2;
fprintf(' %g\n',y)
x = x + 1; % brojač
end
```

Primjer: Varijabli x dodjeljuje se vrijednost 100, a varijabli y vrijednost 1,5. U **while** petlji ispitajte je li vrijednost varijable x veća od 1 ili je vrijednost varijable y manja od 100. Ako je ispunjen taj uvjet, izvršava se dio programa između **while** i **end**, odnosno vrijednost varijable x se dijeli sa 2, a vrijednost varijable y množi sa 3.

```
x = 100
y = 1.5
while (x>1) | (y<100)
x = x / 2
```

```
y = y * 3
end
```

Program unutar **while** petlje izvršio se 7 puta odnosno vrijednost varijable x se 7 puta dijelila sa 2, a vrijednost varijable y se 7 puta množila sa 3. Prilikom petog prolaska kroz **while** petlju varijabla y nije bila manja od 100, ali je varijabla x bila veća od 1. Tek kada ni jedan uvjet nije bio ispunjen, izvršavanje programa se prekida.

M-File funkcije

U MATLAB-u postoji mogućnost kreiranja vlastitih funkcija u formi M-datoteka pohranjenih na računalu. M-file funkcija je slična običnom M-file-u, tekst datoteci s ekstenzijom `.m`

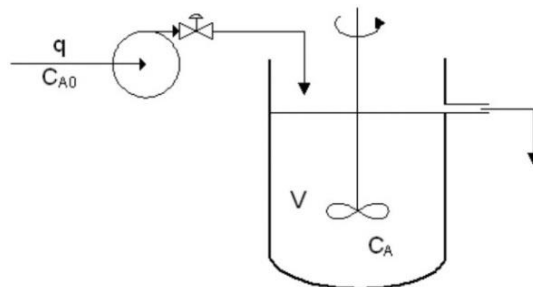
```
function y = ime_funkcije(x1,x2,...,xn)
y = . . .
```

Ime funkcije i ime datoteke moraju biti identični! Npr., funkcija `primjer` mora biti pohranjena u datoteci `primjer.m`

Ako se upiše `primjer` u Naredbeni prozor, MATLAB prvo traži varijablu s tim imenom. Ako je nema, onda je traži kao ugrađenu funkciju. Na kraju provjerava u trenutnom direktoriju postoji li M-datoteka `primjer.m`. Ako se ne nalazi ni u trenutnom direktoriju Matlab pretražuje direktorije koji se nalaze na MATLAB path-u.

Zadatak:

Odredite prijelazni odziv koncentracije u **protočno kotlastom reaktoru**, c_A ako se koncentracija ulazne struje, c_{A0} trenutačno (skokomično) poveća u petoj minuti za $0,92 \text{ mol/m}^3$.



Zadani podaci:

- | | |
|---|--|
| $q = 0,085 \text{ m}^3/\text{min}$ | - protok kroz reaktor |
| $V = 2,1 \text{ m}^3$ | - volumen reaktora |
| $\Delta c_{A,0} = 0,92 \text{ mol/m}^3$ | - iznos skokomične promjene koncentracije tvari A na ulazu u reaktor |
| $c_{A,0} = 1,85 \text{ mol/m}^3$ | - koncentracija tvari A u ulaznoj struji |
| $t_{\text{skok}} = 5 \text{ min}$ | - vrijeme skokomične promjene |

Početna koncentracija tvari A u reaktoru jednaka je koncentraciji na ulazu u reaktor $\rightarrow c_{A,\text{poč}} =$

$$c_{A0} = c_A = 1,85 \text{ mol/m}^3$$

Iz bilance tvari slijedi: $\frac{dn_A}{dt} = \dot{n}_d - \dot{n}_o$

$$V \cdot \frac{dc_A}{dt} = q \cdot c_{A0} - q \cdot c_A$$

Poznato je analitičko rješenje za odziv procesa prvog reda na skokomičnu promjenu ulazne veličine:

$$C_A = C_{A,poc} + \Delta C_A \cdot k \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$$

Pri čemu je $k=1$, a vremenska konstanta $\tau = V/q$

Program:

```
clear all; % brise radni prostor
clc; % cisti ekran

% Zadani procesni podaci
V = 2.1; % volumen reaktora [m3]
q = 0.2; % protok [m3/min]
% V = input('Unesi vrijednost volumena:');
% q = input('Unesi vrijednost protoka:');
caPoc = 1.85; % pocetna koncentracija u reaktoru
deltaca0 = 0.92; % iznos skokomicne promjene

% Podaci za simulaciju
t = 0; % pocetno vrijeme je nula
tend = 120; % konačno vrijeme [min]
delt = 0.5; % vremenski korak
n = (tend-t)/delt; % broj vrem. koraka
tSkok = 5; % trenutak kad nastupa promjena

% SIMULACIJA
for cnt = 1:n % vrem. petlja (od 1 do 120 min, 1 korak 0.5 min)
    t = t + delt;
    ca = caPoc; % pocetna koncentracija u reaktoru

    % konc. u reaktoru nakon skokomicne promjene konc. na ulazu
    if t >= tSkok
        ca = caPoc + deltaca0*(1 - exp(-(t-tSkok)/(V/q)));
    end
    % Pohrana rezultata za graficki prikaz (u obliku matrica)
    CA(cnt) = ca;
    T(cnt) = t;
end

% Graficki prikaz rezultata
plot (T,CA) % odziv (izlaz)
xlabel ('vrijeme, min')
ylabel ('Ca, mol/m3')
title ('Koncentracija u reaktoru Ca')
```

```
fprintf('Koncentracija:\n ')\nfprintf('%.2f \n',CA)
```