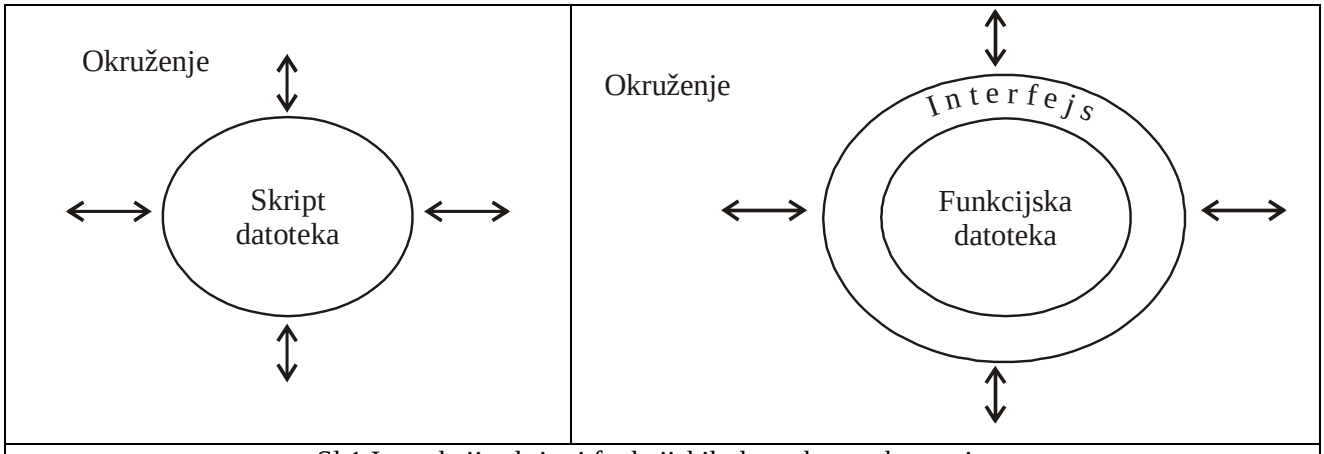
Funkcije

U inzenjerskoj praksi cesto se javljaju tipski problemi. Na primer, potrebno je odrediti sile i momente u clancima resetkastog nosaca, u konzolama izlozenim kontinualnom opterecenju, pritiske u cevovodima razlicitog precnika i duzine, toplotni protok kroz zidove razlicite strukture i sastava, struje i napone u elektricnim kolima razlicite kongiguracije, itd. Pri resavanju takvih problema koriscenjem nekog softverskog alata kao sto je Matlab, cesto se koriste isti skupovi komandi sa istim redosledom izvrsavanja. Razlika je samo u podacima koji se obradjuju takvim naredbama. Da bi smo povecali svoju efikasnost pozeljno je da takve nizove naredbi jednom napisemo i sacuvamo a onda ih koristimo po potrebi. Rad sa povezanim skupom naredbi, koje cine funkcionalnu celinu, u komandnom prozoru Matlaba krajnje je neefikasan. Mnogo jednostavnije resenje je da se takve naredbe sacuvaju u jednom fajlu i izvrsavaju kao celina. U prethodnom poglavlju videli smo da skript datoteke predstavljaju jedno takvo resenje. U ovom poglavlju nesto cemo vise reci o funkcijskim datotekama u kojima se cuvaju funkcije koje definise korisnik, Funkcije koje definise korisnik, ili korisnicke funnkcije, pise sam korisnik da bi ugradio nove mogucnosti u Matlab. One su potpuno ravnopravne kao i ugradjene funkcije kao sto su *abs(x)*, *min(x)*, *sin(x)* itd. Mozemo na primer da napisemo svoju funkciju *koren(x)* i da je ravnopravno koristimo sa funkcijom *sqrt(x)*. Korisnicke funkcije predstavljaju Matlab-ov nacin da se pisu potprogrami koji postoje u drugim programskim jezicima (*Basic*, *C*, *Java*, itd.).

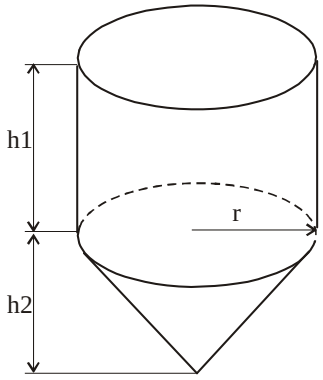
Funkcije predstavljaju kompaktniji i robustniji nacin organizovanja komandi u odnosu na skript fajlove. Osnovna razlika izmedju skript i funkcijskih datoteka je u nacinu komunikacije sa radnim prostorom (Workspace-om) Matlab-a. Izmedju skript fajlova i radnog prostora postoji dvosmerna komunikacija. Svi podaci iz radnog prostora vidljivi su u skript fajlu i svi podaci iz skript fajla, nakon njegovog izvrsenja, vidljivi su i u radnom prostoru. Skript fajlovi prakticno predstavljaju nacin organizovanog izvrsavanja skupa komandi u komandnom prozoru. I izmedju funkcijskih datoteka i radnog prostora postoji dvosmerna komunikacija ali se ona odvija kroz jasno definisani interfejs. Promenljive iz radnog prostora nisu podrazumevano vidljive u funkcijskim datotekama niti su one iz funkcijskih datoteka vidljive u radnom prostoru nakon izvrsenja funkcije. Na slici Sl.1 simbolicki je prikazana interakcija skript i funkcijskih datoteka i Matlab okruzenja.



Sl.1 Interakcija skript i funkcijskih datoteka sa okruzenjem

Funkcijske datoteke su mnogo formalnije u svojoj komunikaciji sa radnim prostorom pa su zato mnogo otpornije na promenu radnog okruzenja u kojem se izvrsavaju. Na primer, zahvaljujuci interfejsu sadrzaj funkcijske datoteke mozemo lakse promeniti a da se to ne odrazi negativno na okruzenje. Ako zelimo da pravimo svoju biblioteku funkcionalnih modula koju mozemo razmenjivati i sa drugima pozeljnije je da se to uradi preko funkcija i odgovarajucih datoteka.

Kreiranje korisnicke funkcije ilustrovacemo jednim primerom. Pretpostavimo da se bavimo projektovanjem i proizvodnjom tipskih rezervoara za tecnost koji se sastoje od cilindricnog i konusnog dela (Sl.2).

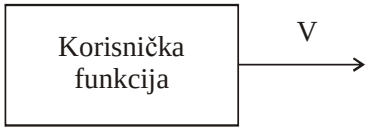


Sl. 2 Prikaz rezervoar sa dimenzijama koje su poznate

U tehnickoj specifikaciji rezervoara potrebno je da prikazemo njegovu zapreminu. Posto se taj zadatak cesto ponavlja napisacemo korisnicku funkciju koja na osnovu poznatih podataka racuna njegovu zapreminu.

Izrada funkcije, u opstem slucaju, nije linearni proces koji se sastoji od niza uzastopnih koraka koji nas od problema vode ka resenju. Pre je to niz koraka koji se ponavlja sve dok kroz nekoliko iteracija ne stignemo do resenja. Prvo definisemo sta funkcija treba da radi a onda i kako da to uradi.

Naravno, na pocetku treba da znamo zbog ceka pisemo funkciju, odnosno koji rezultat ocekujemo od nje. U nasem slucaju to je jasno, zelimo da odredimo zapreminu rezervoara V. Ma kako to cudno izgledalo, u nekim slucajevima nije bas lako jasno definisati sta se ocekuje od funkcije. Medjutim, dok se to ne uradi dalji rad nije moguc. Na slici Sl.3 simbolicki je prikazan razlog pisanja funkcije.



Sl.3 Simbolicki prikaz izlaznih rezultata korisnicke funkcije

Korisnicka funkcija je predstavljena kao "crna kutija" o cijem sadrzaju jos nista ne znamo. Jedino sto znamo je da funkcija treba da nam izracuna zapreminu rezervoara (izlazna strelica). U sledecoj etapi treba definisati podatke koje funkcija ocekuje da bih mogla da nam uradi ono sto zelimo. Iz matematike je poznato da bi smo nasli zapreminu pravilnog valjka i kupe treba da znamo povrsinu njihovih baza i odgovarajuce visine. Posto u nasem slucaju cilindricni i konusni deo rezervoara imaju zajednicku bazu kruznog oblika (Sl.2)

dolazimo do potrebnih ulaznih podataka: *r, h1* i *h2*. Prethodni prikaz funkcije sad mozemo dopuniti ulaznim podacima (Sl.4).

Sl.4 Ulazno-izlazni podaci korisnicke funkcije

Na prethodnoj slici i dalje nemamo nikakvih detalja o unutrasnjosti same funkcije ali imamo informaciju o nacinu njene komunikacija sa spoljnim svetom, odnosno o njenom interfejsu. Na osnovu interfejsa definisemo ugovor sa funkcijom: ako joj obezbedimo ulazne podatke ona treba da na vrati podatak izlazni podatak.

U sledecoj etapi definisemo nacin realizacije ugovora. Treba eksplicitno navesti kako da od podataka o *r, h1* i *h2* stignemo do podatka o velicini *V*. Iz matematike je poznato da se zapremina cilindricnog dela moze izracunati na sledeci nacin:

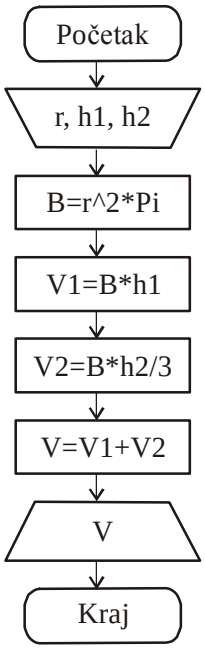
$$V_1 = Bh_1$$

gde je: $B = r^2\pi$ povrsina osnove.

Za zapreminu konusa vazi:

$$V_2 = \frac{1}{3}Bh_2$$

Na osnovu prethodnih izraza mozemo definisati proceduru za odredjivanje zapremine rezervoara sa slike Sl.2. Ta procedura je prikazana u obliku algoritma na slici Sl.5.



Sl.5 Algoritam za odredjivanje zapremine rezervoara

Prethodni algoritam pokazuje sve korake koje je potrebno sprovesti u izracunavanju zapremine. Jasno se vidi gde je pocetak algoritma i sta je potrebno od podataka da bi se odredila zapremina. Prvo se odredjuje povrsina baze koristeci operacije koje Matlab poznaje. Nakon toga odredjuje se zapremina cilindricnog dela a onda i konusnog. Povrsina baze se mora izracunati pre zapremina dok redosled izracunavanja zapremina nije bitan. Ukupna zapremina se dobija sabiranjem prethodno sracunatih zapremina. Ukupna zapremina predstavlja izlaznu velicinu.

Sada kada smo definisali sve korake u realizaciji korisnicke funkcije, nezavisno od programskog jezika koji se koristi, potrebno je napisati odgovarajucu Matlab-ovu funkciju. Medjutim, to sada nije toliko kreativan posao vec je vise stvar prevodjenja jednog idejnog resenja (Sl.5) u nesto sto Matlab razume.

Funkcije se pisu i uredjuju u prozoru koji se otvara iz menija komandnog prozora (*File->New->Function*) a cuvaju se u datotekama koje se zovu funkcijske datoteke. Prvi red funkcijske datoteke mora da pocne sa deklaracijom funkcije koja definise njen interfejs, odnosno nacin njene komunikacije sa spoljnim svetom. Deklaracija korisnicke funkcije ima oblik:

```
function [izlazni_argumenti]=ImeFunkcije(ulazni_argumenti)
```

Na pocetku se nalazi sluzbena rec `function` koja ukazuje da se u datoteci definise funkcija. Ukoliko datoteka pocne nekom drugom recju formirace se skript datoteka. Nakon toga, u uglastim zagradama, navodi se lista izlaznih argumenata odvojenih zarezom. Lista argumenata predstavlja listu simbolickih imena memorijskih lokacija u koje funkcija upisuje rezultate rada. Rezultatima pristupamo pomocu promenljivih pridruzenih tim memorijskim lokacijama. Preko izlaznih argumenata funkcija ispunjava svoju obavezu prema okruzenju. Funkcija moze imati nula, jedan ili vise izlaznih argumenata. Ukoliko postoji samo jedan izlazni argument moze se navesti bez uglastih zagrada. Posle znaka jednakosti navodimo ime funkcije. Ime funkcije biramo sami ali pri imenovanju funkcije moramo postovati pravila koja vaze i pri imenovanju promenljivih. Na primer, u nazivu funkcije ne sme biti razmaka a pravi se razlika izmedju velikih i malih slova. Ime funkcije jednoznacno odredjuje funkciju i mora biti jedinstveno u direktorijumu gde je snimljena funkcijska datoteka. Iza imena, u malim zagradama, navodi se lista ulaznih argumenata funkcije. To je lista promenljivih u koje smestamo podatke koje funkcija koristi da bi odredila rezultat. Preko ulaznih argumenata okruzenje ispunjava svoju obavezu prema funkciji. Kao i kod izlaznih argumenata, moramo voditi racuna o broju i redosledu ulaznih argumenata. Funkcija moze imati nula, jedan ili ulaznih izlaznih argumenata. Ukoliko nema ulaznih argumenata ne moraju se pisati male zagrade. Ukoliko postoji vise argumenata oni se medjusobno odvajaju zarezom. Svaki od argumenata u ulzno/izlaznoj listi moze biti skalar ili niz proizvoljne dimenzije. Deklaraciju funkcije za izracunavanje zapremine rezervoara (Sl.4 i Sl.5) mozemo definisati na sledeci nacin:

```
function [V]=ZapreminaRezervoara(r, h1,h2)
```

Osim preko ulazno/izlaznih argumenata funkcija sa okruzenjem moze komunicirati pomocu naredbi za interaktivni unos `input`, naredbi za prikazivanje podataka na ekranu `disp`, snimanje u fajl `fprintf` ili za crtanje grafika `plot`. Ukoliko pri izracunavanju vrednosti neke promenljive u telu funkcije izostavimo tacku-zarez (;) vrednost te promenljive bice prikazana u komandnom prozoru. Medjutim, funkcija i radni prostor ne dele promenljive. Ono sto je vidljivo u radnom prostoru nije vidljivo u funkciji sem ako se ne

prosledi kao ulazni argument. Slicno, ono sto je vidljivo u funkciji nije vidljivo u radnom prostoru osim ako se ne vrati kao izlazni argument.

Ispod deklaracije funkcije obicno se navodi jedan red komentara (pocinje znakom %) u kojem kratko opisujemo cemu sluzi funkcija. Red se oznacava kao H1 red i mada nije obavezan pozeljno je ga napisati. Posle nekog vremena taj red nam moze pomoci da se setimo sta radi funkcija bez potrebe da detaljno pregledamo kod. Ako u prozoru *Current Folder* kliknemo na datu funkciju pojavice se opis iz reda H1 i bez otvaranja iste funkcije. Ispod reda H1 mozemo otkucati komentar sa detaljnijim opisom funkcije. Taj opis ce se pojaviti kada u komandnom prozoru otkucamo:

```
help ImeFunkcije
```

Nakon komentara unosimo naredbe koje od ulaznih podataka treba da nas odvedu do rezultata koji zelimo. Mozemo koristiti sve naredbe kao u komandnom prozoru. Mozemo koristiti ugradjene funkcije, skript fajlove, druge korisnicki definisane funkcije. Pri pisanju koda treba voditi racuna o mogucoj dimenziji ulaznih podataka. Ukoliko ulazni podaci mogu biti nizovi tada se matematicke operacije mogu izvrsavati koristeci pravila linearne algebre ili se mogu izvrsavati nad pojedinacnim clanovima nizova. U telu funkcije mozemo unositi komentare po potrebi.

Na osnovu algoritma prikazanog na slici Sl.5 funkcijska datoteka sa funkcijom koja racuna zapreminu rezervoara sa slike Sl.2 prikazana je na slici Sl.6.

```
function  [V]=ZapreminaRezervoara (r,h1,h2)
%Funkcija izracunava zapreminu rezervoara koji se sastoji od cilindricnog
%ikonusnog dela
%ulazni argumenti funkcije su:
%r - poluprecnik osnove; h1 - visina cilindricnog dela; h2 - visina
%konusnog dela
    B=r.^2*pi; %povrsina baze
    V1=B.*h1;  %zapremina cilindricnog dela
    V2=B.*h2;  %zapremina konusnog dela
    V=V1+V2;   %ukupna zapremina
 end
```

Sl.6 Funkcija za izracunavanje zapremine rezervoara

Moze se primetiti da smo pri definisanju matematickih operacija predvideli mogucnost da ulazni podaci (r, h1, h2) budu skalari ili vektori.

Nakon sto smo napisali funkciju potrebno je da je snimimo u datoteku. Ukoliko je funkcija slozenija mozda ce biti potrebno da uneti sadrzaj snimimo i pre potpunog zavrsetka funkcije. Cak je i preporucljivo da funkcijsku datoteku snimamo i nakon manjih izmena. Pri prvom snimanju funkcijskog fajla potrebno je da odredimo dve stvari: direktorijum gde cemo snimiti datoteku i naziv pod kojim cemo sacuvati datoteku. Podrazumevano datoteka se snima u tekucem direktorijumu (*Current Folder*). Ako zelimo direktorijum mozemo promeniti pomocu padajuce liste na vrhu prozora za snimanje fajla. Iako se fajl moze snimiti pod razlicitim imenima strogo se preporucuje da se fajlu da isto ime kao sto je naziv funkcije. Funkcija sa slike Sl.6 treba da se snimi u fajl cije je ime: `ZapreminaRezervoara`. Sam Matlab ce fajlu pridruziti

ekstenziju .m. Ukoliko se funkcija snimi u fajl cije ime je razlicito od imena same funkcije tada u pozivu funkcije koristimo ime fajla a ne ime funkcije. Mi pozivamo fajl a Matlab pokrece prvu funkciju koju nadje u tom fajlu.

Nakon sto smo napisali korisnicku funkciju i snimili je u fajl, pozeljno je da je pokrenemo i testiramo. Ulazno/izlazni argumenti u definiciji funkcije predstavljaju formalne parametre. Ulaznim argumentima se pri pozivu funkcije, pre pocetka njenog izvrsavanja, pridruzuju konkretne numericke vrednosti. Sama funkcija izlaznim argumentima pridruzuje numericke vrednosti nakon izvrsavanja funkcije. Sintaksa poziva funkcije ima oblik:

```
[izlazni_argumenti]=ImeFunkcije(ulazni_argumenti)
```

Prvo navodimo listu stvarnih izlaznih argumenata. Obicno je to lista promenljivih u koje funkcija treba da upise rezultate. Lista ulaznih argumenata predstavlja listu konkretnih numerickih vrednosti koja po broju i redosledu odgovara formalnim ulaznim parametrima u deklaraciji funkcije.

Na primer, ako zelimo da izracunamo zapreminu rezervoara cije su dimenzije r=2 [m], h1=4 [m], h2=3 [m], korisnicku funkciju bi smo pozvali:

```
ZR=ZapreminaRezervoara(2,4,3)
```

Pri pozivu funkcije `ZapreminaRezervoara` prvo bi se formalnim parametrima r, h1, h2 pridruzile konkretne numericke vrednosti 2, 4, 3 tacno tim redom. Zatim bi se izvrsile sve operacije u telu funkcije onim redom kako su navedene. Na kraju bi se rezultat upisao u promenljivu ZR. Pri pisanju funkcije matematicke operacije definisemo pomocu memorijskih lokacija a ne pomocu konkretnih podataka. Pri pozivu funkcije nasa je obaveza je da prethodno u te memoriske lokacije upisemo konkretne podatke. Promenljiva na levoj strani znaka jednakosti u pozivu funkcije, ZR u gornjem slucaju, Matlab-u kaze gde treba da upise rezultat. Ukoliko ne navedemo ime te promenljive rezultat ce biti upisan u dezurnu promenljivu `ans`. Pri pozivu funkcije lista ulaznih argumenata mora da odgovara po broju i redosledu ulaznim argumentima koji su navedeni u deklaraciji funkcije.

Na primer, ako prethodnu funkciju pozovemo:

```
ZR=ZapreminaRezervoara(2,4)
```

javice se greska sa upozorenjem da nemamo dovoljno ulaznih argumenata u pozivu. Ako pri pozivu funkcije koristimo tacan broj ulaznih argumenata ali u pogresnom redosledu funkcija ce se izvrsiti ali ce rezultat biti sasvim pogresan. Takav oblik greske je tesko otkriti pa je pri pozivu funkcije potrebno postovati njenu deklaraciju. Na primer, ako funkciju pozovemo naredbom:

```
ZR=ZapreminaRezervoara(4,2,3)
```

promenljiva r dobice vrednost 4, h1 vrednost 2, a h2 vrednost 3. Tako cemo dobiti rezultat koji je razlicit od rezultata u prvom pozivu jer su ulazni podaci navedeni u drugacijem redosledu.

U listi ulaznih argumenata korisnicke funkcije, kao i kod ugradjenih funkcija, mozemo da navodimo promenljive, slozenije izraze, ugradjene funkcije, druge korisnicke funkcije, njihove kombinacije. Na primer mozemo napisati:

```
ZR=ZapreminaRezervoara(sqrt(4),x^2,min(y))
```

Podrazumeva se da smo pre poslednjeg poziva definisali koliko je x i y.

Posto smo u definiciji funkcije koristili operacije nad pojedinacnim elementima, u pozivu funkcije mozemo da koristimo i vektore. Na primer neka je:

```
r=[2 3 4];
h1=[4 2 3];
h2=[2 1 2];
```

Tada mozemo pozvati funkciju:

```
ZR=ZapreminaRezervoara(r,h1,h2)
```

Rezultat ce biti vektor:

```
ZR =
   75.3982   84.8230  251.3274
```

Za ovakve ulazne podatke u nekom drugom programskom jeziku najverovatnije da bi smo morali da napisemo petlju koja bi tri puta pozivala funkciju `ZapreminaRezervoara` sa razlicitim kombinacima ulaznih argumenata $\{(2,4,2);(3,2,1);(4,3,2)\}$. U Matlab-u je dovoljna samo jedna naredba zbog njegove orijentisanosti ka nizovima kao osnovnoj srukturi podataka