2.3.2. Masyvų elementų adresavimas

MATLAB'e skirtingai nuo kitų aukšto lygio programavimo kalbų galima adresuoti ne tik vieną masyvo elementą, tačiau ir bet kokį jų poaibį, t.y. masyvo elementų indeksai gali buti masyvai.

Pavyzdžiui, tarkime, kad *a* yra *n*-elementis vektorius, *b* yra $m \times n$ formato matrica. Tada užrašai: *a*(*5*), *a*(*i*), *b*(2,3), *b*(*i,j*), čia *i* ir *j* – natūralieji skaičiai $i \in [1, n]$, $j \in [1, m]$, tiek MATLAB'e, tiek ir kitoje aukšto lygio programavimo kalboje reikštų tą patį: 5-asis vektoriaus *a* elementas, *i*-sis vektoriaus *a* elementas, matricos *b* elementai b_{23} ir b_{ij}.

Tačiau komandos $x = a(1:5)$, *y*=*b*([1 3], 5:8) reikštų, kad *x* bus masyvas, sudarytas iš pirmųjų 5-ių vektoriaus *a* elementų, o *y* bus 2×4 formato matrica, sudaryta iš matricos *b* elementų, esančių 1-oje ir 3-ioje eilutėse ir 5-me, 6-me, 7-me ir 8-me stulpeliuose. 2.8-oje lentelėje pateiktas vienmačių ir dvimačių masyvų elementų adresavimas

2.8 *lentelė. Masyvų elementų adresavimas*

Adresavimas	*Paaiškinimai*
Vienmačiai masyvai	
a(k)	Adresuojamas *k*-sis masyvo *a* elementas
a(k:l)	Adresuojami masyvo *a* elementai, pradedant *k*-tuoju ir baigiant *l*-tuoju.
a(k:end)	Adresuojami masyvo *a* elementai, pradedant *k*-uoju ir baigiant paskutiniuoju..

a(end:-1:1)	Adresuojami masyvo *a* elementai, pradedant paskutiniuoju ir baigiant pirmuoju. Pavyzdžiui,
	`>> a=[1 2 3 4 5]`
	`>> b=a(end:-1:1)`
	`b = 5 4 3 2 1`

a(x), čia x – galimų indeksų masyvas	Adresuojami tie masyvo *a* elementai, kurių indeksai yra masyvo *x* elementai. Pavyzdžiui,
	`>> a=-3:4`
	`a = -3 -2 -1 0 1 2 3 4`
	`>> c=a([1 2 1 1 3 4])`
	`c = -3 -2 -3 -3 -1 0`

Dvimačiai masyvai	

a(r,p), čia r ir p – matricos a eilučių ir stulpelių indeksų vienmačiai masyvai(aibės)	Adresuojami matricos *a* elementai, esantys *r*-ose eilutėse ir *p*-tuose stulpeliuose, ,t.y.,šių elementų indeksai sutampa su aibių *r* ir *p* Dekarto sandaugos: $r \times p$ elementais.
	Pavyzdžiui,
	`>> a=[1 2 3 4;5 6 7 8;9 10 11 12]`
	`a = 1 2 3 4`
	`5 6 7 8`
	`9 10 11 12`
	`>> b=a([1 2],[1 3 4])`
	`b =`
	`1 3 4`
	`5 7 8`

a(r,:)	Adresuojami matricos *a* elementai, esantys eilutėse, kurias nurodo vienmačio masyvo *r* elementai. Pavyzdžiui, tegu *a*,- prieš tai buvusio pavyzdžio matrica. Tada
	`>> c=a([1 3],:)`
	`c = 1 2 3 4`
	`9 10 11 12`

a(:,r)	Adresuojami matricos *a* elementai, esantys stulpeliuose kuriuos nurodo vienmačio masyvo *r* elementai. Pavyzdžiui,
	`>> a=[1 2 3 4;5 6 7 8;9 10 11 12]`
	`a = 1 2 3 4`
	`5 6 7 8`
	`9 10 11 12`
	`>> d=a(:,[2 4])`
	`d = 2 4`
	`6 8`
	`10 12`

$a_{ij} = a(k)$, *čia a – (m×n) formato matrica, o k – natūralusis skaičius, tenkinantis nelygybę:* $k \le m*n$	Adresuojamas matricos *a* elementas a_{ij} ,čia *i*=*mod*(*k,m*), o *j*=*ceil*(*k/m*). Kitaip tariant, ši komanda ekvivalenti komandoms: *b*=*a*(:); , *aij*=*b*(*k*);. Pavyzdžiui,
	`>> a=[1 2 3 4;5 6 7 8;9 10 11 12]`
	`a = 1 2 3 4`
	`5 6 7 8`
	`9 10 11 12`
	`>> aij=a(8)`
	`aij =`
	`7,`
	nes *i*=*mod*(8,3)=2,o *j*=*ceil*(8/3)=3.

a(x), čia a – vienmatis ar dvimatis masyvas, o x – to paties formato loginis masyvas	Adresuojami tie masyvo *a* elementai, kuriems atitinkami masyvo *x* elementai yra „*true*“. ***Pastaba.*** Masyvas, kuris sudarytas iš elementų „*true*“ (žymimas „1“) arba *„false“* (žymimas „0“), vadinamas loginiu masyvu. Tarkime *x* = `1,3,-5,2,8`. Tada komandos *abs(x) > 2* rezultatas bus loginis masyvas *ans* = `0,1,1,0,1,` t.y. vienetukai (*„true“*) žymi masyvo elementus, kurių moduliai didesni už 2. Vadinasi, komandos *x(abs(x)>2))* rezultatas bus *ans =3, -5 ,8*. Tačiau komandos *ind*=[`0,1,1,0,1`]; *x(ind)*; bus klaidingos, nes *ind*= `0,1,1,0,1` yra ne loginių, bet natūraliųjų skaičių masyvas, o indekso reikšmė „`0`“ MATLAB‘e negalima. Norint gauti teisingą rezultatą, reikia rašyti: *x(logical(ind)).* Funkcija *„logical“* natūraliųjų skaičių masyvą, sudarytą iš „1“ ir „0“ paverčia loginiu masyvu, sudarytu iš elementų „*true*“ ir „*false*“.

Naudojant masyvų adresavimą, o taip pat logines operacijas (jos bus nagrinėjamos vėliau), iš esamų masyvų elementų lanksčiai galime formuoti naujus masyvus.

Elementų šalinimas iš vektorių ir matricų. Lankstus masyvų elementų adresavimas įgalina lengvai šalinti vektorių ir matricų elementus.

Norėdami pašalinti vektoriaus elementus, jiems priskiriame nulinio formato matricą — „[]“.

Pavyzdys. Panagrinėkime komandas:

```
>> x=1:8
x =  1     2     3     4     5     6     7     8
>> x(3)=[] % šaliname 3-ią elementą
x =  1     2     5     6     7     8
>> x(2:5)=[] % šaliname elementus, pradedant 2-ju ir baigiant 5-ju
x =  1     7     8
>> a=1:5
a =  1     2     3     4     5
>> a(end-2:end)=[] % šaliname 3-is paskutinius elementus
a =  1     2
```

Norėdami pašalinti matricos elementus, elgsimės analogiškai, tačiau turėsime šalinti ***visus*** matricos stulpelio ar eilutės elementus. Šalinti ***pavienius*** matricos elementus ***negalima*** , nes operacijos rezultatas privalo būti stačiakampė matrica.

Pavyzdys. Aptarsime komandas:

```
      >> a=[1 2 3;4 5 6;7 8 9]
a =  1     2     3
     4     5     6
     7     8     9
>> a(:,2)=[]  % šaliname 2-ąjį matricos stulpelį
a =  1     3
     4     6
     7     9
>> a(3,1)=[]
???  Indexed empty matrix assignment is not allowed.
```

(Elementui negalima priskirti tuščiosios matricos).

2.3.3. Operacijos su masyvais.

Aptarsime operacijas su masyvais. Pradžioje aptarsime skaliaro ir masyvo operacijas.

Skaliaro ir masyvo operacijos. *Atliekant masyvo ir skaliaro sudėties, atimties, daugybos ir dalybos operacijas, nagrinėjamoji operacija vykdoma tarp kiekvieno masyvo elemento ir skaliaro.*

Pavyzdžiui, jei a = 1, 3, -5, 4 yra vienmatis masyvas, tai atliekant veiksmą *b=a+2* gausime: b=3, 5, -3, 6, t.y. dvejetas pridedamas prie kiekvieno masyvo *a* elemento.

Jei c =1 2 3
4 5 6,

tai, atlikus komandą, 2*c-1, turėsime matricą : 1 3 5
7 9 11.

Masyvo ir masyvo operacijos. Su masyvais (vektoriais, matricomis) galima atlikti matematikoje įprastas operacijas: sudėtį, atimtį ir daugybą. Aišku, kad masyvų formatai turi būti tinkami: sudėčiai ir atimčiai matricų formatai turi būti tokie patys, o, dauginant matrica *a* iš matricos *b,* matricos *a* stulpelių skaičius turi būti lygus matricos *b* eilučių skaičiui.

Be paminėtų operacijų, MATLAB‘as leidžia atlikti masyvų ***paelementes*** (***element-by-element***) operacijas. Šios operacijos žymimos matematikoje įprastu veiksmo ženklu, prieš kuri dedamas taškas. Aišku, kad anksčiau aptartos masyvo ir skaliaro operacijos yra paelementės. Paelementės operacijos vienmačiams masyvams surašytos 2.9 lentelėje. Dvimačiams masyvams – matricoms – šios operacijos yra analogiškos.

2.9 *lentelė. Masyvų paelementės operacijos*

Operacijos	*Operacijos turinys*
	$a=[a_1,a_2,\cdots,a_n]$, Duomenys: $b=[b_1,b_2,\cdots,b_n]$, $c-skaliaras$.
$a\pm c$	$a\pm c=[a_1\pm c,a_2\pm c,\cdots,a_n\pm c]$
$a*c$	$a*c=[a_1*c,a_2*c,\cdots,a_n*c]$
a/c	$a/c=[a_1/c,a_2/c,\cdots,a_n/c]$
$a\pm b$	$a\pm b=[a_1\pm b_1,a_2\pm b_2,\cdots,a_n\pm b_n]$
$a.*b$	$a.*b=[a_1*b_1,a_2*b_2,\cdots,a_n*b_n]$
$a./b$	$a./b=[a_1/b_1,a_2/b_2,\cdots,a_n/b_n]$
$a.\hat{}c$	$a.\hat{}c=[a_1.\hat{}c,a_2.\hat{}c,\cdots,a_n.\hat{}c]$
$c.\hat{}a$	$c.\hat{}a=[c.\hat{}a_1,c.\hat{}a_2,\cdots,c.\hat{}a_n]$
$a.\hat{}b$	$a.\hat{}b=[a_1.\hat{}b_1,a_2.\hat{}b_2,\cdots,a_n.\hat{}b_n]$

Be išnagrinėtų operacijų, MATLAB‘as leidžia atlikti matricų *a* ir *b* Kronekerio tenzorinę sandaugą, kurią realizuoja funkcija *kron(a,b)*. Ši funkcija sukuria blokinę matricą kiekvieną matricos *a* elementą padaugindama iš matricos *b*.

Pavyzdžiui,

```
>> a=[1 2;3 4]
a =  1     2
     3     4
>> b=[1 2 3;4 5 6]
b =  1     2     3
     4     5     6
>> c=kron(a,b)
c =  1     2     3     2     4     6
     4     5     6     8    10    12
     3     6     9     4     8    12
    12    15    18    16    20    24
```

2.3.4. Masyvų elementų rikavimas.

Sprendžiant uždavinius, dažnai tenka masyvų elementus išrikiuoti didėjimo arba mažėjimo tvarka. Tam tikslui MATLAB‘e yra funkcija *sort.*

Tarkime, kad *x* yra vienmatis masyvas. Tada operacija *sort(x*) masyvo *x* elementus išrikiuoja didėjimo tvarka. Funkcija *sort* turi du išėjimo parametrus: *xs* ir *idx.* Parametras *xs* yra masyvas, sudarytas iš masyvo *x* elementų, išrikiuotų didėjimo tvarka, o parametras *idx* yra šių elementų indeksai (adresai) masyve *x*.

Pastaba. Jei MATLAB‘o funkcija turi kelis išėjimo parametrus, tai jie rašomi laužtiniuose skliaustuose lygybės ženklo kairėje pusėje. Vadinasi, [*xs,idx*]=*sort*(*x*) rezultatas bus anksčiau apibūdinti masyvai.

Pavyzdys. Panagrinėsime komandas:

```
>> x=randperm(8)
x =  4     8     5     3     6     2     7     1
>> a=sort(x)
a =  1     2     3     4     5     6     7     8
>> [a,ind]=sort(x)
a =  1     2     3     4     5     6     7     8
```

```
ind = 8     6     4     1     3     5     7     2
```

Masyvo *ind* pirmasis elementas reiškia, kad masyvo *a* *i*-ojo elemento adresas masyve *x* yra *ind*(*i*). Pvz., *ind*(1) = 4 reiškia, kad *a*(1) masyve *x* stovi 4-oje vietoje.

Pastaba. Funkcija *sort* turi dar vieną įėjimo parametrą,- funkcijos vardą, kuris gali įgauti reikšmes: *descend* arba *ascend* ir šie vardai rašomi tarp apostrofų.

Jei įrašysime *'descend',* tai masyvas bus rikiuojamas mažėjimo tvarka, o jei įrašysime *'ascend'* tai masyvas bus rikiuojamas didėjimo tvarka.

Norint masyvo *x* elementus išrikiuoti mažėjimo tvarka nenurodant parametro *descend*, elgsimės taip:

1. išrikiuosime juos didėjimo tvarka, t.y. *xs*=*sort*(*x*),
2. atliksime komandą *xsd*=*xs*(*end*:-1:1) arba *xsd*=*rot*90(*xs,2*).

Tai reiškia, kad pirmuoju atveju masyvo *xs*, išrikiuoto didėjimo tvarka, elementus perrašysime atvirkščia tvarka, t.y. pradedant paskutiniuoju ir baigiant pirmuoju, o antruoju atveju – masyvą *xs* pasuksime 180 laipsniu kampu prieš laikrodžio rodyklę.

Panagrinėkime dvimačio masyvo – matricos *a* rūšiavimą . Šiuo atveju komandos

[*as,adx*]= *sort*(*a, 'ascend'*)

rezultatas bus dvi matricos:

- *as,* kurioje kiekviename stulpelyje bus patalpinti matricoa *a* atitinkamų stulpelių elementai, išrikiuoti didėjimo tvarka,
- *idx* – šių elementų eilučių numeriai matricoje *a.*

Pavyzdys. Atlikime komandas:

```
>> a=[randperm(4);randperm(4);randperm(4);randperm(4)]
a =  2     3     1     4
     3     1     2     4
     1     4     3     2
     1     4     3     2
>> [as,ind]=sort(a,'ascend')
 % tą patį rezultatą turėsime ir su komandą: [as,ind]=sort(a)
as = 1     1     1     2
     1     3     2     2
     2     4     3     4
     3     4     3     4
ind =
     3     2     1     3
     4     1     2     4
     1     3     3     1
     2     4     4     2
```

Be aptartos komandos, matricoms yra dar dvi rūšiavimo funkcijos: *sort*(*a,*1) ir *sort*(*a,*2). Funkcija *sort*(*a,*1) sutampa su funkcija *sort*(*a*), t.y. atskirai rikiuojamas kiekvienas matricos *a* stulpelis. Funkcija *sort*(*a,*2) atskirai rikiuoja kiekvieną matricos *a* eilutę.

Pastaba. Kaip minėjome, funkcija *sort* turi du išėjimo parametrus: išrikiuotą masyvą *as* ir indeksų masyvą *ind*. Kreipiantis į *sort,* nebūtina nurodyti abu išėjimo parametrus ,- paprastai nurodomas tik pirmasis išėjimo parametras.

2.3.5 Submasyvų formavimas ir funkcijos find, max ,min, sum ir prod.

Dažnai tenka ieškoti masyvų elementų, tenkinančių nurodytas sąlygas, pavyzdžiui, rasti teigiamus masyvo elementus. MATLAB'e šis uždavinys sprendžiamas funkcijos *find* pagalba.

Funkcija find. Funkcijos *find* įėjimo parametras yra loginė išraiška, o išėjimo parametrai yra indeksai, žymintis tuos įėjimo masyvo elementus, kuriems loginė išraiška yra teisinga. Pavyzdžiui, norėdami rasti masyvo *x* teigiamus elementus, rašysime komandą:
k=find (x>0);.

Pavyzdys.

```
>> x=[10 -2 13 -4 -5 8]
x = 10    -2    13    -4    -5     8
>> k=find(x>0)
k = 1     3     6 % masyvo x teigiamų elementų indeksai
>> a=x(k)
a = 10    13     8 % masyvo x teigiami elementai.
```

Funkcija *find* operuoja ir su dvimačiais masyvais – matricomis.

Pavyzdys.

```
>> a=[1 2 3;4 5 6;7 8 9]
a =  1     2     3
     4     5     6
     7     8     9
>> [i,j]=find(a>6)
i =  3
     3
     3

j =  1
     2
     3,
```

čia *i* ir *j* atitinkamai rodo numerius eilučių ir stulpelių, kurių susikirtime yra matricos *a* elementai, didesni už 6. Masyvų *i* ir *j* elementai rodo, kad matricos *a* elementai a_{31}, a_{32}, a_{33} yra didesni už 6.

Funkcija *find* ,matricos atveju, gali grąžinti ir vienmatį indeksų masyvą.

Pavyzdys.

```
>> k=find(a>6)
k =  3
     6
     9.
```

Komanda "`k=find(a>6)`" analogiška komandoms:

```
>> b=a(:)
b =  1
     4
     7
     2
     5
     8
     3
     6
     9
>> k=find(b>6)
k =  3
     6
     9.
```

Komanda "`b=a(:)`" transformuoja matricą *a* į stulpelį *b,* o komanda "`k=find(b>6)`" nurodo stulpelio *b* elementų, didesnių už 6, indeksus.

Funkcijos max ir min. Šalia funkcijos *find* labai dažnai naudojamos funkcijos *max* ir *min,* kurios atitinkamai randa masyvo didžiausius ir mažiausius elementus.
Pavyzdys.

```
>> x=[10 -2 13 -4 -5 8]
x = 10    -2    13    -4    -5     8
>> mn=max(x)
mn = 13
>> [mx,ind]=max(x)
mx = 13
ind =
     3
```

Paskutinės komandos rezultatai rodo, kad masyvo *x* didžiausias elementas yra 13, ir jis yra 3-iasis masyvo elementas.

Norėdami rasti masyvo visų didžiausių elementų indeksus, elgsimės taip, kaip parodyta žemiau pateiktame pavyzdyje:

```
>> y=[1 4 6 3 2 1 6]
y = 1     4     6     3     2     1     6
>> my=max(y)
my = 6
>> k=find(y==my)
k = 3     7
```

Masyvas *y* turi du didžiausius elementus, kurių indeksai yra 3 ir 7.

Dvimačio masyvo – matricos atveju *max* ir *min* funkcijos veiks truputį kitaip.

Panagrinėkime pavyzdį:

```
>> a=randperm(12)
a = 3    6   11    5    2    8    7    4    9    1   12   10
>> a=reshape(a,3,4)
a =  3     5     7     1
     6     2     4    12
    11     8     9    10
>> [ma,ind]=max(a)
ma = 11     8     9    12
ind =
     3     3     3     2
```

Matome, kad masyvo *ma* elementai yra matricos *a* kiekvieno stulpelio didžiausi elementai, o masyvo *ind* elementai nurodo numerius eilučių, kuriose šie elementai yra.

Norėdami rasti didžiausią matricos elementą, turėsime rašyti komandą:

```
>> d=max(max(a))
d = 12
arba
>> d=max(ma)
d = 12.
```

Norėdami rasti didžiausią matricos elementą ir jo buvimo vietą, galime elgtis taip:

```
>> [da,ind]=max(a(:))
da = 12
ind =
    11.
```

Šiuo atveju pirmiausia matricą transformuojame į stulpelį, o po to, randame šio stulpelio didžiausią elementą ir jo indeksą.

Pastaba. Tiek vienmačio, tiek ir dvimačio masyvų atvejais analogiškai veikia ir funkcija *min*.

Funkcijos sum ir prod. Skaitiniuose metoduose labai patogios funkcijos: *sum(x)* ir *prod(x)*, iš kurių pirmoji skirta rasti masyvo *x* elementų sumą, o antroji – sandaugą.

Jei *x* yra vienmatis masyvas , tai

$sum(x) = \sum_{i=1}^{n} x_i$, o $prod(x) = \prod_{i=1}^{n} x_i$.

Jei *x* yra $m \times n$ matrica, tai *y=sum(x)* grąžina *n*-elementį masyvą *y*, kurio *j*-asis elementas yra matricos *x* *j*-ojo stulpelio elementų suma. Analogiškai, komandos *z=prod(x)* rezultatas bus vienmatis masyvas *z*, kurio *j*-asis elementas yra matricos *x* *j*-ojo stulpelio elementų sandauga.

Nesunku suvokti, kad komandos *sum*(*sum*(x)) arba *sum*(*x*(:)) apskaičiuos matricos *x* visų elementų sumą, o - *prod*(*prod*(x)) arba *prod*(*x*(:)) apskaičiuos matricos *x* visų elementų sandaugą.

Funkcijos cumsum ir cumprod. Be minėtų ***sum*** ir ***prod*** funkcijų MATLAB‘e yra sumas bei sandaugas kaupiančios funkcijos: ***cumsum*** ir ***cumprod.***

Vektoriui *x* komandos *y= cumsum*(*x*) rezultatas yra *x* formato vektorius *y*, kurio *i*-oji komponentė yra vektoriaus *x* visų komponenčių nuo pirmosios iki *i*-os imtinai suma, t.y. $y_i = \sum_{j=1}^{i} x_j$.

Analogiškai veikia ir funkcija *cumprod.* Tik šiuo atveju vektoriaus *y* *i*-oji komponentė yra vektoriaus *x* visų komponenčių nuo pirmosios iki *i*-os imtinai sandauga, t.y. $y_i = \prod_{j=1}^{i} x_j$.

Jei *x* yra matrica, tai komandos *y= cumsum*(*x*) rezultatas yra to paties formato matrica *y*, kurios stulpelių elementai yra atitinkamų matricos *x* stulpelių kaupiamosios sumos. Aišku, kad komandos *y= cumprod*(*x*) rezultatas yra analogiškas. Tik šiuo atveju matricos *y* stulpelių elementai yra atitinkamų matricos *x* stulpelių kaupiamosios sandaugos.

Pavyzdys.

```
>>x=randperm(6)
x = 2  4  3  6  5  1
>>cs=cumsum(x)
cs = 2  6  9  15  20  21
>>cp=cumprod(x)
cp = 2  8  24  144  720  720
>>a=[1 2 3; 5 6 2; 1 -2 -4]
a =  1     2     3
     5     6     2
     1    -2    -4
 >>sa=cumsum(a)
sa = 1     2     3
     6     8     5
     7     6     1

 >>pa=cumprod(a)
 pa = 1     2     3
      5    12     6
      5   -24   -24
```

2.3.5 Masyvų formatų radimo funkcijos

MATLAB‘e yra eilė funkcijų, skirtų rasti masyvo elementų skaičių, o taip pat masyvo formatą. Šios funkcijos aprašytos 2.10 lentelėje.

2.10 *lentelė.* *Masyvo formatų funkcijos*

Funkcijos	*Aprašymas*
s=size(*a*)	Šios funkcijos išėjimo parametras yra vektorius *s,* kurio pirmoji komponentė parodo masyvo *a* eilučių, o antroji – stulpelių skaičių.
[*r,c*]=*size*(*a*)	Ši komanda grąžina du skaliarus: *r* ir *c,* iš kurių pirmasis yra masyvo *a* eilučių, o antrasis – stulpelių skaičius.
r=size(*a,1*)	Skaliaras *r* yra masyvo *a* eilučių skaičius.
c=size(*a,2*)	Skaliaras *c* yra masyvo *a* stulpelių skaičius
n=length(*a*)	Ši komanda grąžina: *max*(*size*(*a*)), jei *a* – netuščiasis masyvas, 0, jei *a* – tuščiasis masyvas, vektoriaus elementų skaičių, jei *a* – vektorius.
n=numel(*a*)	Ši komanda grąžina masyvo *a* elementų skaičių.

Pavyzdys.

```
>> a=[1 2 3 4;5 6 7 8]
a =  1     2     3     4
     5     6     7     8
>> s=size(a)
s =  2     4
>> [r,c]=size(a)
r = 2
c = 4
>> r=size(a,1)
r = 2
>> c=size(a,2)
c = 4
>> numel(a)
ans = 8
>> length(a(:))
ans = 8
>> length(a)
ans = 4
>> max([r,c])
ans = 4.
```

2.4 Santykiai ir loginės operacijos

Be anksčiau išnagrinėtų įprastinių matematinių operacijų MATLAB‘e, kaip ir kitose aukšto lygio programavimo kalbose, naudojamos santykio ir loginės operacijos.

Santykio operacijos. Santykio operacijos patalpintos 2.11 lentelėje.

Santykio operacijos rezultatas yra loginė konstanta: „*true*“ arba „*false*“.MATLAB‘e, skirtingai nei kitose aukšto lygio programavimo kalbose, santykio operandai gali būti to paties formato masyvai. Tada ir santykio operacijos rezultatas yra to paties formato loginis masyvas, t,y, masyvas, sudarytas iš loginių konstantų: „*true*“ (ši konstanta žymima 1) ir „*false*“ (ši konstanta žymima 0). Loginiai masyvai yra specifiniai masyvai, kurie gali būti naudojami tiek, kaip loginiai masyvai, tiek adresavimui, tiek ir aritmetinėse išraiškose, kaip skaičiai.

2.11 *lentelė.* *Santykio operacijos*

Operacijos ženklas	*Paaiškinimai*

<	Mažiau
<=	Mažiau arba lygu
>	Daugiau
>=	Daugiau arba lygu
==	Lygu
~=	Nelygu.

Pavyzdžiai.

```
>> a=1:5, b=5-a
a = 1     2     3     4     5
b = 4     3     2     1     0
>> t=a>2
t = 0     0     1     1     1
>> a==b
ans = 0     0     0     0     0
>> c=a(t) %loginis masyvas t naudojamas adresuoti masyvo a elementus
c = 3     4     5
>> x=[-3:3]/3
x = -1.0000   -0.6667   -0.3333     0    0.3333    0.6667    1.0000
>> x=x+(x==0)*eps %loginis masyvas naudojamas kaip skaitinis masyvas
x = -1.0000  -0.6667  -0.3333   0.0000   0.3333   0.6667   1.0000
>> sin(x)./x %naudodami suformuotą masyvą x, išvengiame dalybos iš 0
ans = 0.8415   0.9276   0.9816   1.0000   0.9816   0.9276   0.8415
```

Loginės operacijos. Loginių operacijų (loginių išraiškų) operandai yra loginės konstantos, loginiai kintamieji ir loginės išraiškos. Tačiau MATLAB‘e, skirtingai nei kitose aukšto lygio programavimo kalbose, loginių operacijų bei loginių išraiškų operandai gali būti ne tik loginiai kintamieji, bet ir jų masyvai. Be to, MATLAB‘as bet kokį skaičių, nelygu 0, traktuoja kaip „*true*“, o skaičių, lygu 0, - kaip „*false*“.

Loginės operacijos patalpintos 2.12 lentelėje.

2.12 *lentelė. Loginės operacijos*

Loginė operacija	*Paaiškinimai*
&	Operacija „ir“ (loginė sandauga): 0&0=0, 0&1=0, 1&0=0, 1&1=1.
|	Operacija „arba“ (loginė suma): 0|0=0, 0|1=1, 1|0=1, 1|1=1.
~	Neigimo operacija (inversija); ~0=1, ~1=0.

Pavyzdžiai.

```
>> a=1:9, b=9-a
a = 1     2     3     4     5     6     7     8     9
b = 8     7     6     5     4     3     2     1     0
>> t=a>4
t = 0     0     0     0     1     1     1     1     1
>> tf=~(a>4)
tf = 1     1     1     1     0     0     0     0     0
>> (a>2)&(a<6)
ans = 0     0     1     1     1     0     0     0     0
>> x=-pi:1:pi
x = -3.1416 -2.1416 -1.1416 -0.1416  0.8584  1.8584  2.8584
>> y=sin(x)
y = -0.0000 -0.8415 -0.9093 -0.1411  0.7568  0.9589  0.2794
```

```
>> z=(y>0).*y % loginis masyvas traktuojamas kaip skaitinis
z = 0    0    0    0    0.7568    0.9589    0.2794
```

Paskutinioji komanda suformuoja masyvą *z,* kuriame išlieka masyvo *y* teigiamos reikšmės, o vietoje neigiamų *y* reikšmių stovi nuliai.

Operacijų eiliškumas. 2.13 lentelėje surašytas operacijų vykdymo eiliškumas, pradedant operacijomis, turinčiomis aukščiausiąjį prioritetą, ir baigiant operacijomis, turinčiomis žemiausią prioritetą.

2.13 *lentelė. Operacijų eiliškumas*

Operacijos	*Prioritetas*
Skliaustai "( )"	Aukščiausias
Paelementis transponavimas: „ .' ", transponavimas : „ ' „ , paelementis kėlimas laipsniu: „ .^", matricų kėlimas laipsniu: „^"	
Unarinis pliusas: „+", unarinis minusas : „-",neigimas: „~"	
Paelementė daugyba: „.", matricų daugyba: „*", paelementė dalyba: kairioji „./" arba dešinioji „.\", matricų dalyba: kairioji „/" arba dešinioji „\".*	
Sudėtis: „+", atimtis: „-"	
Stulpelio operacija: „(:)", mažiau negu: „<", mažiau arba lygu: „<=", daugiau: „> ", daugiau arba lygu: „>=", lygu: „==", nelygu: „~="	
Loginė sandauga: „&"	
Loginė suma: „|"	Žemiausias

Loginės funkcijos. Be aptartų santykio bei loginių operacijų, MATLAB'as turi eilę loginių funkcijų. 2.14 lentelėje yra pateiktos pagrindinės MATLAB'o loginės funkcijos.

2.14 *lentelė. Loginės funkcijos*

Funkcijos	*Aprašymas*
xor(x,y)	Suma moduliu du: *xor*(0,0)=0, *xor*(0,1)=1, *xor*(1,0)=1, *xor*(1,1)=0.
any(x)	*any(x)=true,* jei bent vienas masyvo *x* elementas nelygus 0; *any(x)=false,* jei visi masyvo *x* elementai lygūs 0.
all(x)	*all(x)=true,* jei visi masyvo *x* elemetai lygūs 1; *all(x)=false,* jei bent vienas masyvo elementas lygus 0.
isreal(x)	*isreal(x)=true,* jei *x* yra realusis skaičius arba realiųjų skaičių masyvas; *isreal(x)=false,* priešingu atveju.
isnan(x)	*isnan(x)=true,* jei *x =nan;* *isnan(x)=false,* priešingu atveju.
isinf(x)	*isinf(x)=true,* jei *x=inf*; *isinf(x)=false,* priešingu atveju.
isprime(x)	*isprime(x)=true,* jei *x* – pirminis skaičius; *isprime(x)=false,* priešingu atveju.
isempty(x)	*isempty(x)=true,* jei *x*=[ ]; *isempty(x)=false,* priešingu atveju.

Skaičiai nan ir tuštieji masyvai. MATLAB‘e išskirtinę reikšmę turi skaičiai *nan* (*„not a number“*) ir tuštieji masyvai: „[ ]“. Pagal IEEE matematikos standartus, beveik visų operacijų, kuriose dalyvauja skaičiai *nan,* rezultatas yra *nan.* Tačiau MATLAB‘e, su skaičiais *nan* atliekant santykio operacijas , galima gauti netikėtą rezultatą, nes atskiri *nan* tarpusavyje nėra lygūs.

Pavyzdys.

```
>> a=[1 2 nan inf nan]
a = 1     2   NaN   Inf   NaN
>> b=2*a
b = 2     4   NaN   Inf   NaN
>> c=(a==nan)
c = 0     0     0     0     0
>> d=(a~=nan)
d = 1     1     1     1     1.
```

Paskutinųjų dviejų komandų rezultatai yra truputį netikėti ir rodo, kad atskiri *nan* tarpusavyje nėra lygūs.

Tuštieji masyvai yra MATLAB’o kūrėjų išmįslas, žymintis, kad masyvas, neturitis elementų,- yra tuščias.

Panagrinėkime kelis pavyzdžius.

```
>> size([])
ans = 0     0
>> a=zeros(0,5)% masyvas, turintis 5-is tuščiuosius stulpelius
a = Empty matrix: 0-by-5
>> size(a)
ans = 0     5
>> d=ones(4,0) % masyvas, turintis 4-ias tuščiąsias eilutes
d = Empty matrix: 4-by-0
>> size(d)
ans = 4     0
>> length(d) % masyvo ilgis lygus nuliui, nors jis turi 4-ias
               tuščiąsias eilutes
ans = 0
>> x=-2:2
x = -2    -1     0     1     2
>> y=find(x>2)
y = Empty matrix: 1-by-0 % masyve x nėra nei vieno elemento,
                  didesnio už 2, todėl masyvas y yra tuščiasis
>> isempty(y)
ans = 1
```

2.5 Sąlyginio valdymo ir ciklo operatoriai

Skaičiavimo procesui valdyti MATLAB‘e, kaip ir kitose aukšto lygio programavimo kalbose, yra *sąlyginio valdymo ir ciklo operatoriai.* Tačiau šie MATLAB‘o operatoriai yra „galingesni“.

Sąlyginio valdymo „if - else – end“ operatorius. Paprasčiausias sąlyginio valdymo operatorius yra:

if loginė išraiška
komandos
end.

Komandos tarp žodžių *if* ir *end* bus vykdomos, jei visi *loginės išraiškos* elementai yra *true*.

Bendru atveju *loginė išraiška* gali būti ir *algebrinė išraiška.* Tada:

- jei *algebrinės išraiškos* visi elementai yra nelygūs nuliui, tai jos reikšmė bus traktuojama kaip *true;*

- jei *algebrinės išraiškos* bent vienas elementas bus lygūs nuliui, tai jos reikšmė bus suprantama kaip *false.*

Tačiau patartina taip „piratiškai“ nesielgti. Todėl toliau, rašydami *loginė išraiška,* ją suprasime kaip *loginį masyvą.* Be to, sakydami *loginė išraiška = = true,* suprasime, kad visi loginio masyvo elementai yra *true*.

Jei norime, kad nagrinėjamo sąlyginio valdymo operatoriaus *komandos* būtų vykdomos, kai bent vienas *loginės išraiškos* elementas yra *true*, tai reikia naudoti loginę funkciją *any.* Primename, kad funkcijos *any*(*loginė išraiška*) rezultatas yra *true,* jei bent vienas *loginio masyvo* elementas yra *true*.

Pastaba*:* Jei *loginė išraiška* sudaryta iš kitų loginių išraiškų, sujungtų loginėmis operacijomis, tai MATLAB‘as, apskaičiuodamas *loginės išraiškos* reikšmę, skaičiuoja galimai mažesnį *loginių išraiškų* skaičių.

Pavyzdžiai*.*

loginė išraiška= loginė išraiška1 | loginė išraiška2.

Jei *loginė išraiška1=true,* tai *loginė išraiška2*,- neskaičiuojama, nes ,šiuo atveju *loginė išraiška = = true* nepriklausomai nuo *loginės išraiškos2* reikšmės.

Analogiškai,

loginė išraiška = loginė išraiška1 & loginė išraiška2.

Jei *loginė išraiška1=false,* tai *loginė išraiška2*,- neskaičiuojama, nes šiuo atveju *loginė išraiška = = false* nepriklausomai nuo *loginės išraiškos2“* reikšmės.

Pilnasis sąlyginio valdymo sakinys yra:

```
if  loginė išraiška
    Komandos, kurios vykdomos, kai loginė išraiška = = true
else
     Komandos, kurios vykdomos, kai loginė išraiška = = false
end;
```

Jei yra trys ar daugiau alternatyvų, tai sąlyginio valdymo komanda užrašoma:

```
if  loginė išraiška 1
    Komandos, kurios vykdomos, kai loginė išraiška1= = true
elseif  „loginė išraiška 2“
     Komandos, kurios vykdomos, kai loginė išraiška 2= = true
elseif  loginė išraiška 3
     Komandos, kurios vykdomos, kai loginė išraiška = = true
elseif  loginė išraiška 4
          .  .  .
 else
    Komandos, kurios vykdomos, kai visos anksčiau išvardintos
     loginės išraiškos yra false
end;
```

for tipo ciklas. *for* ciklas užrašomas:

```
for x=masyvas
   komandos
end;
```

Tarp žodžių *for* ir *end* parašytos *komandos* bus vykdomos po vieną kartą kiekvienam masyvo *x* stulpeliui.

Pavyzdžiai*.*

```
1.  for n=1:10
```

```
      y(n)=sin(n*pi/10);
    end
    y
y = 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090 0.5878 0.3090
0.0000
```

Bus skaičiuojamos funkcijos *sin*(*n*pi/10*) reikšmės, kai *n* kinta nuo 1 iki 10 kas žingsnį 1.

```
2. i=1;
  x=rand(4,5)
  for t=x
      y(i)=sum(t);
      i=i+1;
  end
  y
x = 0.0579    0.1389    0.2722    0.4451    0.8462
    0.3529    0.2028    0.1988    0.9318    0.5252
    0.8132    0.1987    0.0153    0.4660    0.2026
    0.0099    0.6038    0.7468    0.4186    0.6721
y =
    1.2338    1.1442    1.2331    2.2616    2.2462
```

Pirmiausia, pasinaudodami atsitiktinių skaičių generavimo procedūra *rand*, generuojame 4×5 formato matricą *x*. Po to, ciklo *for* pagalba skaičiuojame šios matricos kiekvieno stulpelio elementų sumą.

Abu šie pavyzdžiai aiškina ciklo *for* veikimą, tačiau jie nėra efektyvaus programavimo MATLAB'u pavyzdžiai. *for* ciklas neturėtų būti naudojamas, jei, sprendžiant uždavinį, galima pasinaudoti operacijomis su masyvais. ***Toks sprendimo būdas vadinamas sprendimo vektorizacija.***

Vektorizacija yra kodo, operuojančio su skaliarais, transformacija į kodą, operuojantį su vektoriais. Paprastai transformuotas kodas, operuojantis su vektoriais ir matricomis, yra aiškesnis, nei pradinis kodas, kuris operuoja su skaliarais ir naudoja *for* arba *while* tipo ciklus. Aišku, kad vektorizacija nekeičia skaičiaus aritmetinių operacijų, reikalingų rezultatui apskaičiuoti; ji tik keičia operacijų atlikimo pobūdį: operacijas su skaliarais keičia operacijomis su vektoriais bei matricomis, kurios realizuotos MATLAB'o branduolyje ir atliekamos greičiau nei skaliarinės operacijos interpretuojamos *m*-faile. Todėl vektorizuotas kodas yra įvykdomas greičiau, nei ekvivalentiškas kodas, operuojantis su skaliarais.

Pirmąjį pavyzdį išspęstume efektyviau, jei kodą vektorizuotume taip:

```
>> n=1:10;
>> y=sin(n*pi/10)
y = 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511 0.8090 0.5878 0.3090
0.0000,
```

o antrąjį pavyzdį efektyviau išspręstume kodą vektorizuodami taip:

```
>> x=rand(4,5);
>> y=sum(x)
y = 1.2338    1.1442    1.2331    2.2616    2.2462.
```

Aptarkime dar vieną vektorizavimo pavyzdį,- apskaičiuokime *n*-elemenčio masyvo *x* gretimų elementų sumas, kurias patalpinkime į masyvą *b*.

Aišku, kad matematiškai šio uždavinio sprendimo algoritmas bus užrašomas taip: $b_i = x_i + x_{i+1}, i = 1,2,\ldots,n-1$.

Matlabo kodas, realizuojantis šį algoritmą, būtų toks:

```
n=length(x);
b=zeros(1,n-1);
```

```
for i=1:n-1
    b(i)=x(i)+x(i+1);
end
```

Pirmoji kodo komanda nustato masyvo *x* elementų skaičių, o antroji išskiria atminties vietą masyvui *b,* t.y. masyvas *b* užpildomas nuliais. Kai jau išskirta masyvui *b* atminties vieta*,* tai "*for* " ciklas veikia greičiau, nes kiekvienam naujam elementui *b* nebereikia išskirinėti atminties.

Algoritmo MATLAB'o kodas, panaudojus operacijas su masyvais, bus ne tik trumpesnis, bet ir efektyvesnis:

```
b=x(1:end-1)+x(2:end);
```

Reikia pabrėžti, kad šie pavyzdžiai nerodo, kad "*for* " ciklą reikia šalinti iš matlabo kodų. Jie tik parodo, kad, kur galimos operacijos su masyvais, jas ten ir reikia naudoti. Kai "*for* "ciklai naudojami tinkamai, jie yra patogūs, greiti ir efektyvūs.

While tipo ciklas. *While* tipo ciklas užrašomas taip:

while loginė išraiška
komandos
end;

Komandos, parašytos tarp žodžių *while* ir *end,* bus vykdomos, jei ***visi logonės išraiškos elementai bus true***.

Pastaba: Visa, kas buvo paskyta apie *loginę išraišką,* nagrinėjant operatorių *if-else-end,* tinka ir ciklo *while-end* operatoriui.

Pavyzdys. Apskaičiuokime mažiausią skaičių, kurį pridėjus prie vieneto, gautume skaičių, didesni už 1, t.y. apskaičiuokime anksčiau aptartą MATLAB'o konstantą ***eps***.

```
num=0; epsilon=1;
while (1+epsilon)>1
    epsilon=epsilon/2;
    num=num+1;
end
epsilon=epsilon*2;
num=-num
eps

epsilon = 2.2204e-016,
num = -53,
ans = 2.2204e-016.
```

Kaip matome, apskaičiuotas skaičius ir MATLAB'o konstanta *eps* sutampa. Parametras *num* rodo, neigiamą laipsnį, kuriuo pakėlus 2, gautume ieškomą skaičių.

Switch – case operatorius. Šis operatorius naudojamas, kai, priklausomai nuo parametro reikšmės, yra atliekamas vienas iš kelių galimų skaičiavimo variantų. *Swich-case* operatoriaus struktūra yra:

switch išraiška
case testo išraiška1
komandos1
case {testo išraiška2, testo išraiška3, testo išraiška4}
komandos2
otherwise
komandos3
end;

Šiame operatoriuje *išraiška* gali būti arba *skaliaras*, arba simbolių *eilutė* (*string*).

Jei *išraiška* yra skaliaras, tai tikrinamas santykis: *išraiška==testo išraiškaN* kiekvienai *case* testo išraiškai, t.y. *testo išraiška1, testo išraiška2, testo išraiška3 ir t.t.* Vykdoma ta grupė komandų, kurioms šis santykis yra *true.*

Jei išraiška yra simbolių eilutė, tai *išraiška==testo išraiškaN* tikrinama funkcijos *strcmp*(*išraiška,testo išraiškaN*) pagalba.

Pastaba*: Funkcija strcmp*(*s1,s2*) *duoda rezultatą* ***true****, jei simbolių eilutės* ***s1*** ir ***s2*** *yra lygios; ir – rezultatą* ***false***, *- priešingu atveju*.

Pateiktam s*wich-case* operatoriui turėsime:

- jei *išraiška==testo išraiška1* yra *true,* tai vykdomos *komandos1;*
- jei *išraiška==testo išraiška2* | *išraiška==testo išraiška3* | *išraiška==testo išraiška4,* tai bus vykdomos *komandos2;*
- jei *išraiška* nebuvo lygi nei vienai anksčiau išvardintai *testo išraiškai,* tai bus vykdomos *komandos3*.

Pavyzdys.

```
>> x=2.7;
>> units='m';
>> switch units
case {'inch','in'}
y=x/2.54;
case {'feet','ft'}
y=x/2.54/12;
case{'meter','m'}
y=x/100;
case {'millimeter','mm'}
y=x*10;
case {'centimeter','cm'}
y=x;
otherwise
disp(['Nežinoma dimensija:',units])
y=nan
end
>> y
y = 0.0270
```

Šios komandos dydį *x*=2,7 *cm* verčia metrais. Suteikus kintamajam *units* kitą reikšmę, pvz. *inch*, centimetrai bus verčiami coliais. Kitaip tariant, priklausomai nuo *units* reikšmės, centimetrai bus keičiami arba coliais, arba pėdomis, arba metrais, arba milimetrais, arba bus pranešama, kad nurodyta *units* reikšmė nežinoma ir rezultatas yra *nan.*

Operatorius try-catch. Šis operatorius vartotojui duoda galimybę tikrinti, kaip MATLAB‘as reaguoja į klaidas. Operatoriaus struktūra yra:

try
  komandos1
catch
  komandos2
end.

Atliekant *try-catch* komandą, yra vykdomos komandos, surašytos *komandos1* bloke.

Jei , vykdant šias komandas, klaidų nerandama, tai *try-catch* komandos vykdymas baigiamas;

Jei , vykdant *komandos1* bloko komandas, randama klaida, tai tolesnis šio bloko komandų vykdymas nutraukiamas ir valdymas perduodamas *„komandos2“* bloko komandoms. Jei tarp šių komandų yra kreipinys į funkciją *lasterr (paskutinė klaida),* tai detaliai bus paaiškintas rastos klaidos pobūdis.

Pavyzdys.

```
x=ones(4,2); y=4*eye(3);
try
    z=x*y;
    z=z+2;
catch
    z=nan
    lasterr
end
z =
   NaN
ans =
Error using ==> *
Inner matrix dimensions must agree.
```

Pirmiausia bus generuojamos matricos;

$$x = \begin{matrix}1 & 1\\ 1 & 1\\ 1 & 1,\end{matrix} \text{ ir } y = \begin{matrix}4 & 0 & 0\\ 0 & 4 & 0\\ 0 & 0 & 4\end{matrix}.$$

Toliau, atliekama komanda `z=x*y;` , tai yra matricos *x* ir *y* – dauginamos. Kadangi (4× 2) ir (3× 3) formato matricų daugyba negalima (klaida), tai valdymas perduodamas komandoms: `z=nan,lasterr`. Pirmoji priskirs *z*-ui *nan,* o antroji - nurodo rastos klaidos pobūdį: *„turi atitikti vidiniai matricų formatai”.*

Komanda break. Komanda *break* yra skirta nutraukti *while – end* arba *for ... end* ciklus. Įvykdžius šią komandą, ciklas nutraukiamas ir valdymas perduodamas pirmai po ciklo einančiai komandai.

Pavyzdys.

```
n=5; y=zeros(1,n); x=rand(1,n)
k=1;
while k <= n
  if x(k) > 0.8
      break
  end
  y(k)=x(k); k=k+1;
end
y
x = 0.7621    0.4565    0.0185    0.8214    0.4447
y = 0.7621    0.4565    0.0185         0         0
```

Ciklas *while – end* bus nutrauktas, kai *k* <= *n* arba *x* masyve bus rastas elementas, didesnis už 0.8.

Komanda return. Jei komanda ***break*** yra skirta nutraukti ***while*** arba ***for*** ciklus, tai komanda ***return*** yra skirta nutraukti funkcijos (*m-failo*) vykdymą. Valdymas perduodamas komandai, esančiai po šią funkciją iššaukusios komandos.

Jei komandos ***break*** pavyzdžio programos tekstą apiformintume kaip funkciją pvz., vardu ***demoreturn*** ir žodį „***break***“ pakeistume žodžiu „***return***“, tai būtų nutrauktas tos funkcijos darbas.

```
function y=demoretun(n)
 y=zeros(1,n);
 x=rand(1,n)
 k=1;
 while k <= n
  if x(k) > 0.8
     return
  end
 y(k)=x(k);
 k=k+1;
end
```

Jei *x* masyve bus rastas elementas didesnis už 0.8, tai funkcijos ***demoreturn*** darbas bus nutrauktas anksčiau, negu bus išnagrinėti visi masyvo *x* elementai,

Komanda pause. Ši komanda laikinai sustabdo funkcijos vykdymą. Įvykdžius šią komandą, programos vykdymas sustabdomas ir komandų lango apačioje pasirodo užrašas: *„Paused: Press any key“* (Pauzė: Paspausk bet kurį klavišą).

Į šią situaciją vartotojas gali reaguoti dvejopai:

- nutraukti programos darbą vienu metu paspausdamas *Ctrl* ir *C* klavišus,
- pratęsti programos darbą, paspausdamas bet kurį, pvz., *return*, klavišą.

Pavyzdys.

```
function y=nonegp(x)
% nonegp grąžina x, jei x>=0; priešingu atveju spausdina pranešimą,
% "x - neigiamas" ir daroma pauzė.
if x < 0
    disp('x neigiamas, paspauskite bet kuri klavišą')
    pause
end
y=x
>> x=-4
x = -4
>> y=nonegp(x)
x neigiamas, paspauskite bet kurį klavišą.
```

Komanda keyboard. Ši komanda veikia panašiai, kaip ir komanda *pause*, tačiau yra “galingesnė”. Įvykdžius šią komandą , funkcijos (*m-failo*) vykdymas sustabdomas, prieš varneles („>>“) atsiranda raidė *K* ir visas valdymas perduodamas vartotojui. Vartotojas turi galimybę sužinoti funkcijos, kurioje patalpinta ši komanda, vidinių kintamųjų reikšmes tam, kad nustatytų klaidos priežastį. Funkcijos (*m-failo*) darbas bus atnaujintas, kai vartotojas į displėjų(komandų langą) įves komandą ***„return“***.