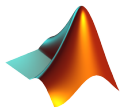


Programování v GNU Octave a Matlab

Programování F1400 + F1400a

doc. RNDr. Petr Mikulík, Ph.D.

podzimní semestr 2020



```
x = 81.0;           % vstupni hodnota x
presnost = 1e-8;    % pozadovana presnost vypoctu

a = x; % pocatecni odhad odmocniny cisla x
do
    aold = a;           % predchozi hodnota
    e = (x/a - a) * 0.5; % odhad chyby
    a = a + e;          % novy odhad
    rel = abs(aold-a) / a;
    fprintf('%.15g %g %g\n', a, e, rel);
until (rel <= presnost);
fprintf('odmocnina z %g je %g\n', x, a);
```

Proměnné – vytváření, rušení

- Proměnné není třeba deklarovat, lze do nich rovnou zapsat hodnotu typu skalár, vektor, matice, řetězec, struktura, identifikátor souboru, ...
- Zápisem do proměnné se zruší její předchozí hodnota a typ, proměnná zůstává až do případného zrušení příkazem `clear`
- Seznam proměnných se vypíše příkazy `who` anebo `whos`.

Příklad:

```
octave> a = 1.2; v = [1, 2, 5]; z = [1, 2; 3, 4];
```

```
octave> who; whos
```

Variables in the current scope:

a v z

Variables in the current scope:

Attr	Name	Size	Bytes	Class
====	====	====	=====	=====
	a	1x1	8	double
	v	1x3	24	double
	z	2x2	32	double

Total is 8 elements using 64 bytes

```
octave> clear v z
```

```
octave> who
```

Variables in the current scope:

a

Proměnné – typy

- Číselné proměnné jsou standardně typu double (reálné číslo ve dvojnásobné přesnosti, tj. 8 B). Jako celočíselná se bere tehdy, je-li její reálná část nulová.

```
octave> a = 3.0
a = 3
octave> b = 4.56
b = 4.5600
```

```
octave> fprintf('a - vystup int %i double %g\n', a, a);
a - vystup int 3 double 3
```

```
octave> fprintf('b - vystup int %i double %g\n', b, b);
b - vystup int 4.56 double 4.56
```

- Speciální účely (čtení velkých matic apod.): lze definovat celočíselné typy int8, uint8, int16, uint16, int32, uint32, int64, uint64.*

```
octave> u = uint16([9 8 7 6])
u =
    9    8    7    6
```

```
octave> whos u
```

Attr	Name	Size	Bytes	Class
====	====	====	=====	=====
	u	1x4	8	uint16

Přístup k prvkům vektoru (pole)

Přístup k prvkům

```
octave> a = [11:15]
```

```
a =
```

```
11 12 13 14 15
```

```
octave> b = a(2:4)
```

```
b =
```

```
12 13 14
```

```
b = a([1 2 5])
```

```
b =
```

```
11 12 15
```

```
octave> c = a(4:end), d = a(4:length(a))
```

```
c =
```

```
14 15
```

```
d =
```

```
14 15
```

```
octave> e = a(1:2:end), f = a(2:2:end)
```

```
e =
```

```
11 13 15
```

```
f =
```

```
12 14
```

Změna prvků

```
octave> a = [11:15]
```

```
a =
```

```
11 12 13 14 15
```

```
octave> a(1) = 111
```

```
a =
```

```
111 12 13 14 15
```

```
octave> a(2:4) = 0
```

```
a =
```

```
111 0 0 0 15
```

```
octave> a(2:4) = []
```

```
a =
```

```
111 15
```

```
octave> a(6) = 666
```

```
a =
```

```
111 15 0 0 0 666
```

```
octave> a(3:5) = [-3 -4 -5]
```

```
a =
```

```
111 15 -3 -4 -5 666
```

Přístup k prvkům matice

Přístup k prvkům

```
octave> a = [1 2 3; 4 5 6]
```

```
a =
```

1	2	3
4	5	6

```
octave> a(2,3)
```

```
ans = 6
```

```
octave> b = a(2, :), c = a(:, 2)
```

```
b =
```

4	5	6
---	---	---

```
c =
```

2
5

```
octave> d = a(1:2, 2:3)
```

```
d =
```

2	3
5	6

```
octave> e = a(:)'
```

```
e =
```

1	4	2	5	3	6
---	---	---	---	---	---

Změna prvků

```
octave> length(a), size(a)
```

```
ans = 3
```

```
ans =
```

2	3
---	---

```
octave> a(:, 2) = -1
```

```
a =
```

1	-1	3
4	-1	6

```
octave> a(3,3) = 42
```

```
a =
```

1	-1	3
4	-1	6
0	0	42

```
octave> a(2,:) = []
```

```
a =
```

1	-1	3
0	0	42

Programování cyklů a podmínek

Cyklus for

```
for k=1:10
    x=10*k
end

for k=0:2:10; k, end

for k=[1 2 10]
    k, 10*k
end

a = [1 2 10]
for k=a
    k, 10*k
end
```

Podmínka if

```
x = 7
if (x == 0)
    fprintf('x je nula');
    fprintf('nula je hezke cislo');
elseif (x==1)
    fprintf('vysledek je 1');
elseif (x==2)
    fprintf('vysledek vysel dva');
    fprintf('ale to se mi nezda');
else
    fprintf('vyslo velke cislo');
end
```

Cyklus while

```
k = 0
while (k < 10)
    8*k
    k++;
end
```

Cyklus do

```
k = 0
do
    k++;
    8*k
until (k == 10)
```

Program na výpočet odmocniny

Program, který jsme již vytvořili v Céčku, přepíšeme do m pro Octave/Matlab:

Program v C

```
1 #include <stdio.h>
2 #include <math.h>
3 int main () {
4     double x, presnost; // vstupni hodnoty
5     x = 81.0;
6     presnost = 1e-8;

7     double a, aold; // iterace vysledku
8     double e, rel; // abs. a rel. chyba
9     a = x; // pocatecni odhad
10    do {
11        aold = a; // predchozi
12        e = (x/a - a)*0.5; // odhad chyby
13        a += e; // novy odhad
14        rel = fabs(aold-a) / a;
15        printf("%.15g %g %g\n",a,e,rel);
16    } while (rel > presnost);
17    // vysledek je v a
18    printf("odmocnina(%g) = %g\n",x,a);
19    return 0;
20 }
```

Program v m

```
1 x = 81.0;           % vstupni hodnoty
2 presnost = 1e-8;

3 a = x; % pocatecni odhad
4 do
5     aold = a; % predchozi
6     e = (x/a - a)*0.5; % odhad chyby
7     a += e; % novy odhad
8     rel = abs(aold-a) / a;
9     fprintf('%.15g %g %g\n',a,e,rel);
10 until (rel <= presnost);
11 % vysledek je v a
12 fprintf('odmocnina(%g) = %g\n',x,a);
```

Efektivní (před)alokování polí

Před použitím je vhodné maticím alokovat dostatek paměti:

Ukázka s vektory

```
1 clear
2 n = 190000;
3 tic
4 for k=1:n
5     a(k) = 2*k+1; % stále se prodlužuj
6 end
7 toc

8 tic
9 b = zeros(1,n); % zaber dostatek místa
10 for k=1:n
11     b(k) = 2*k+1;
12 end
13 toc
```

Spotřebovaný čas:

Elapsed time is 0.589936 seconds.
Elapsed time is 0.554478 seconds.

Ukázka s maticemi

```
1 clear
2 nn = 2000;
3 tic
4 for k=1:nn
5     c(k,k) = 2*k+1; % stále se zvětšuj
6 end
7 toc

8 tic
9 d = zeros(nn,nn); % zaber paměť
10 for k=1:nn
11     d(k,k) = 2*k+1;
12 end
13 toc
```

Spotřebovaný čas:

Elapsed time is 2.64365 seconds.
Elapsed time is 0.0120101 seconds.

Sumy a statistika

- Statistické (i jiné) výpočty lze provádět **cyklem** nebo **vektorově**.
- Příklad: součet prvků pole

```
1 % Vytvoření tří vektorů:
2 x = linspace(0,10);
3 y1 = sqrt(x);
4 y2 = 2*x.x-1;

5 % Zabudovaný příkaz pro součet prvků vektoru:
6 sum(x)
7 sum(y1)
8 sum(y2)

9 % Součet prvků vektoru cyklem:
10 sx=0; for k=1:length(x) sx=sx+x(k); end
11 sy1=0; for k=1:length(x) sy1=sy1+y1(k); end
12 sy2=0; for k=1:length(x) sy2=sy2+y2(k); end

13 % Maticově:
14 % Vytvoření matice ze tří vektorů (řádkové vs sloupcové):
15 a = [x; y1; y2]'
16 % Zabudovaný příkaz pro součet sloupců matice:
17 sum(a)
```

- Viz též funkce min(), max(), mean(), std(), aj.