Cvičení 3

Modelování systémů a procesů

Mgr. Lucie Kárná, PhD

karna@fd.cvut.cz

March 15, 2017

m-files

= příkazy uložené v textovém souboru s příponou `.m`

Typy m-souborů

scripty sekvence příkazů

- všechny proměnné globální
- volají se jménem souboru

m-files

= příkazy uložené v textovém souboru s příponou `.m`

Typy m-souborů

scripty sekvence příkazů

- všechny proměnné globální
- volají se jménem souboru

m-funkce funkce

- všechny proměnné lokální
- vstupní a výstupní parametry
- volají se jménem funkce a parametry
- jméno souboru musí být totožné se jménem funkce

Funkce

`function [y1,y2] = fce(x1,x2)` *<příkazy>* `end`

funkce v `.m` souboru `fce.m`

Funkce

`function [y1,y2] = fce(x1,x2)` *<příkazy>* `end`

funkce v `.m` souboru `fce.m`

Příklad 1

Naprogramujte funkci dvou proměnných $geom(x_1, x_2) = \sqrt{x_1 \cdot x_2}$.

Funkce

`function [y1,y2] = fce(x1,x2)` *<příkazy>* `end`

funkce v `.m` souboru `fce.m`

Příklad 1

Naprogramujte funkci dvou proměnných $geom(x_1, x_2) = \sqrt{x_1 \cdot x_2}$.

Řešení

- v editoru napíšeme

```
function y = geom( x1, x2 )
y = sqrt( x1 * x2 );
end
```

Funkce

`function [y1,y2] = fce(x1,x2)` *<příkazy>* `end`

funkce v `.m` souboru `fce.m`

Příklad 1

Naprogramujte funkci dvou proměnných $geom(x_1, x_2) = \sqrt{x_1 \cdot x_2}$.

Řešení

- v editoru napíšeme

```
function y = geom( x1, x2 )
y = sqrt( x1 * x2 );
end
```

- uložíme pod jménem `geom.m`

Funkce

`function [y1,y2] = fce(x1,x2)` *<příkazy>* `end`

funkce v `.m` souboru `fce.m`

Příklad 1

Naprogramujte funkci dvou proměnných $geom(x_1, x_2) = \sqrt{x_1 \cdot x_2}$.

Řešení

- v editoru napíšeme

```
function y = geom( x1, x2 )
y = sqrt( x1 * x2 );
end
```

- uložíme pod jménem `geom.m`
- voláme z *command line* např. `» a = geom( 7, 11 )`

for-cyklus

`for i=1:n` *<příkazy>* `end`

cyklus, u kterého je předem známý počet opakování

for-cyklus

`for i=1:n` *<příkazy>* `end`

cyklus, u kterého je předem známý počet opakování

Příklad 2

Naprogramujte funkci pro výpočet faktoriálu $n!$.

for-cyklus

`for i=1:n` *<příkazy>* `end`

cyklus, u kterého je předem známý počet opakování

Příklad 2

Naprogramujte funkci pro výpočet faktoriálu $n!$.

Řešení

```
function fn = faktorial( n )
fn = 1;
for i = 1:n
fn = fn * i;
end
end
```

while-cyklus

`while` *<podmínka>* *<příkazy>* `end`

cyklus, kde není předem známý počet opakování

while-cyklus

`while` *<podmínka>* *<příkazy>* `end`

cyklus, kde není předem známý počet opakování

Příklad 3

Naprogramujte funkci, která pro dvě přirozená čísla najde zbytek po dělení prvního čísla druhým.

while-cyklus

`while` *<podmínka>* *<příkazy>* `end`

cyklus, kde není předem známý počet opakování

Příklad 3

Naprogramujte funkci, která pro dvě přirozená čísla najde zbytek po dělení prvního čísla druhým.

Řešení

```
function zb = zbytek ( velke, male )
zb = velke;
while zb >= male
zb = zb - male;
end end
```

Větvení

```
if <podmínka> <příkazy> end
if <podmínka> <příkazy1> else <příkazy2> end
```

Větvení

```
if <podmínka> <příkazy> end
if <podmínka> <příkazy1> else <příkazy2> end
```

Příklad 4

```
function rozhodni ( a, b )
if a>0 disp('a je kladne')
if a==b disp('cisla se rovnaji')
else disp('cisla se nerovnaji')
end
```

for-cyklus

Příklad 5

Jaký typ cyklu použijeme v případě, že je naším úkolem sečíst kvadráty číselných hodnot ve vektoru?

for-cyklus

Příklad 5

Jaký typ cyklu použijeme v případě, že je naším úkolem sečíst kvadráty číselných hodnot ve vektoru?

Řešení

Předem známe počet opakování (délka vektoru), půjde o **for** cyklus.

for-cyklus

Příklad 5

Jaký typ cyklu použijeme v případě, že je naším úkolem sečíst kvadráty číselných hodnot ve vektoru?

Řešení

Předem známe počet opakování (délka vektoru), půjde o **for** cyklus.

```
function s = sumQ( v )
```

for-cyklus

Příklad 5

Jaký typ cyklu použijeme v případě, že je naším úkolem sečíst kvadráty číselných hodnot ve vektoru?

Řešení

Předem známe počet opakování (délka vektoru), půjde o **for** cyklus.

```
function s = sumQ( v )
n = length( v );
s = 0;
for i = 1 : n
s = s + v(i) * v(i);
end
end
```

while-cyklus

Příklad 6

Diferenční rovnice $y[n+1] = y[n]/2$ s počáteční podmínkou $y[0] = 10$. Úkol: najít hodnotu n, pro niž je $y[n] < \varepsilon = 0{,}001$.

Iterace:

- $y[0] = 10$
- $y[1] = y[0]/2 = 5$
- $y[2] = y[1]/2 = 2.5$
- $y[3] = y[2]/2 = 1.25$
- ...

while-cyklus

Příklad 6

Diferenční rovnice $y[n+1] = y[n]/2$ s počáteční podmínkou $y[0] = 10$. Úkol: najít hodnotu n, pro niž je $y[n] < \varepsilon = 0{,}001$.

Iterace:

- $y[0] = 10$
- $y[1] = y[0]/2 = 5$
- $y[2] = y[1]/2 = 2.5$
- $y[3] = y[2]/2 = 1.25$
- ...

Řešení

Neznáme počet opakování $\Rightarrow$ **while**.

while-cyklus

Příklad 6

Diferenční rovnice $y[n+1] = y[n]/2$ s počáteční podmínkou $y[0] = 10$. Úkol: najít hodnotu n, pro niž je $y[n] < \varepsilon = 0{,}001$.

Iterace:

- $y[0] = 10$
- $y[1] = y[0]/2 = 5$
- $y[2] = y[1]/2 = 2.5$
- $y[3] = y[2]/2 = 1.25$
- ...

Řešení

Neznáme počet opakování $\Rightarrow$ **while**.

```
function n = mylim( y0, eps)
```

while-cyklus

Příklad 6

Diferenční rovnice $y[n+1] = y[n]/2$ s počáteční podmínkou $y[0] = 10$. Úkol: najít hodnotu n, pro niž je $y[n] < \varepsilon = 0{,}001$.

Iterace:

- $y[0] = 10$
- $y[1] = y[0]/2 = 5$
- $y[2] = y[1]/2 = 2.5$
- $y[3] = y[2]/2 = 1.25$
- ...

Řešení

Neznáme počet opakování $\Rightarrow$ **while**.

```
function n = mylim( y0, eps)
n = 0; y = y0;
while y >= eps
y = y/2; n = n + 1;
end
end
```

Klouzavý průměr

- průměr konstantního počtu za sebou jdoucích hodnot
- slouží k vyhlazení časové řady

Klouzavý průměr

- průměr konstantního počtu za sebou jdoucích hodnot
- slouží k vyhlazení časové řady

Úkol

- naprogramovat funkci `y = ravg( x, w )`
  - `x` vektor vstupních dat
  - `w` šířka okna (počet průměrovaných hodnot)
  - `y` výstup

Klouzavý průměr

- průměr konstantního počtu za sebou jdoucích hodnot
- slouží k vyhlazení časové řady

Úkol

- naprogramovat funkci `y = ravg( x, w )`
  - `x` vektor vstupních dat
  - `w` šířka okna (počet průměrovaných hodnot)
  - `y` výstup
- zobrazit x a y v grafu v různých barvách

Klouzavý průměr

- průměr konstantního počtu za sebou jdoucích hodnot
- slouží k vyhlazení časové řady

Úkol

- naprogramovat funkci `y = ravg( x, w )`
  - `x` vektor vstupních dat
  - `w` šířka okna (počet průměrovaných hodnot)
  - `y` výstup
- zobrazit x a y v grafu v různých barvách
- k průměrování použít funkci `mean()`
- počáteční hodnoty pro $i < w$ zkopírovat ze vstupu
- vstup: dopravní data
  `http://zolotarev.fd.cvut.cz/static/msap/data.mat`

Klouzavý průměr – řešení

Funkce v souboru `ravg.m`

```
function y = ravg( x, w)
y = x;
for j = w : length( x )
i = j - w + 1;
y(j) = mean( x(i:j) );
end
```

Klouzavý průměr – řešení

Funkce v souboru ravg.m

```
function y = ravg( x, w)
y = x;
for j = w : length( x )
i = j - w + 1;
y(j) = mean( x(i:j) );
end
```

Vykreslení výsledku

```
» load data.mat
» y = ravg ( data, 10 );
» plot( data );
» hold on;
» plot(y,'r','linewidth',2)
```

Simulace hodu férovou mincí s pravděpodobností panna-orel 50–50

Úkol

Naprogramovat funkci `[s,p,o]=coin(n)` simulující n hodů mincí. Výstupem je

- vektor (řetězec) s obsahující symboly `'P'` a `'O'`,
- celé číslo p udávající, kolikrát padla panna, a
- celé číslo o udávající, kolikrát padl orel.

Hod mincí – řešení

Řešení

```
function [s,p,o] = coin( n )
s = char ( zeros ( 1, n ));
p = 0, o = 0;
for i = 1:n
r = rand();
if r < 0.5
s(i) = 'P'; p = p + 1;
else
s(i) = 'O'; o = o + 1;
end end
```