1 Riadenie toku programu

Príkazy v Matlabe na kontrolu toku programu fungujú veľmi podobne ako v iných programovacích jazykoch. Zoznam príkazov:

`IF (IF-END, IF-ELSE-END, IF-ELSEIF-ELSE-END),`
`SWITCH-CASE`
`FOR cykly`
`WHILE cykly`

1.1 FOR cykly

FOR cykly je vhodné použiť, ak vieme presne špecifikovať počet opakovaní cyklu. Syntax je nasledovná:

`for i=` *pole*
príkazy
`end`

Všimnite si, že príkaz `if` je ukončený slovíčkom `end` a nie sú potrebné žiadne zátvorky na vymedzenie tela cyklu.

Príklad 1.1 *Chceme vypísať hodnoty funkcie sínus v bodoch* $\pi n/10$ *pre* $n = 0, 1, ..., 10$. *Vygenerujte tieto hodnoty použitím príkazu* `for`.

Riešenie:

```
for n=0:10
    x(n+1)=sin(pi*n/10);
end
```

Dajte si na záver vypísať finálnu hodnotu vektora x. *Všimnite si tiež, že sme dopredu nikde nedefinovali, akú dĺžku bude mať vektor* x, *a sme ho proste pri každej iterácii nafukovali o ďalšiu zložku. Dĺžka vektora* x *tak postupne narastala od prvej zložky až po konečnú dĺžku.*

1.2 WHILE cykly

WHILE cyklus je vhodné použiť v prípade, že ukončenie opakovania cyklu podmieňujeme porušením zvolenej podmienky, a teda počet opakovaní cyklu nie je vopred známy. Príkazy v tele cyklu sa vykonávajú, pokiaľ je zadaná podmienka splnená.Syntax tohto príkazu je nasledovná:

`while` *podmienka*
príkazy
`end`

Príklad 1.2 *Predpokladajme, že číslo* π *vydelíme dvomi. Výsledné číslo opäť vydelíme dvomi. Tento proces opakujeme, až pokiaľ nedostaneme číslo, ktoré je menšie alebo rovné* 0.01. *Koľko iterácií na to potrebujeme?*

Riešenie:

```
k=1;
cislo=pi;
while cislo>0.01
    cislo=cislo/2;
    k=k+1;
end
```

1.3 Príkaz if-else-end

Matlab umožňuje robiť na základe podmienok vetvenie programu. Syntax je nasledovná:

```
if výraz
     príkazy
else
     príkazy
end
```

Ak je potrebné rozlíšiť viacero alternatív, mala by sa použiť nasledujúca konštrukcia:

```
if výraz1
     príkazy (vykonajú sa, ak je výraz1 pravdivý)
elseif výraz2
     príkazy (vykonajú sa, ak je výraz2 pravdivý)
⋮
else
     príkazy (vykonajú sa, ak sú všetky predchádzajúce výrazy nepravdivé)
end
```

1.4 Príkaz switch-case

Ak je potrebné rozlíšiť veľa alternatívnych diskrétnych možností pre hodnotu danej premennej, je vhodné namiesto príkazu `if` použiť príkaz `switch`. Jeho syntax je nasledovná:

```
switch výraz (skalár alebo string)
     case hodnota1
      príkazy (vykonajú sa, ak sa výraz vyhodnotí na hodnotu1)
     case hodnota2
      príkazy (vykonajú sa, ak sa výraz vyhodnotí na hodnotu2)
⋮
otherwise
     príkazy
end
```

Keď sa Matlab dostane na príslušný case, ďalšie príkazy sa už nevykonávajú, switch sa ukončí. Ak je potrebné rozlíšiť viac ako jednu hodnotu v danej alternatíve, použijeme `{}`.

Príklad 1.3 *Priraďte premennej* `a` *ľubovoľnú hodnotu a otestujte funkčnosť nasledujúcej konštrukcie:*

```
switch a
    case 0
    disp('a=0')
    case {1,2,3,4}
    disp('a je menej ako 5')
    case 10
    disp('a=10')
otherwise
    disp('a je mimo rozsah')
end
```

1.5 Príkaz break

`Break` je užitočný príkaz, ktorý nám umožňuje predčasne ukončiť vykonávanie tela cyklu. Väčšiou sa používa v rámci príkazu `if`.

Príklad 1.4 *V tomto prípade sa ukončí vykonávanie príkazu* `while`*, ak hodnota premennej* `a` *bude menšia ako* 5*.*

```
a=20;
while a>2
    a=a/2;
    if(a<5)
        break;
    end
end
```

2 m-súbory

Slúžia na písanie príkazov rovnako ako v príkazovom riadku (angl. Command line), akurát že sa celá postupnosť príkazov dá uložiť pod jedným menom ako samostatný súbor. Názov súboru nesmie obsahovať medzeru, bodku, ani pomlčku. Prípona matlabovského súboru je ".m". Súbory, s ktorými chceme pracovať a vyvolávať ich, musia byť uložené vo vašom nastavenom pracovnom adresári (angl. Current Directory).

Existujú dva typy m-súborov: *súbor so skriptom* a *súbor s funkciou*. Súbory so skriptom nemajú žiadne vstupné ani výstupné argumenty, súiab jednoduchým zoznamom za sebou idúcich príkazov. Súbory s funkciou môžu mať vstupné alebo výstupné argumenty.

Na vytvorenie m-súboru kliknite na `File`→`New`→`m-file` alebo na ikonku nového súboru. Otvorí sa okno editora. Tu môžete písať svoj kód. Súbor uložíte pomocou `File`→`Save as`. Overte si, či sa súbor uloží do adresára, ktorý máte nastavený ako pracovný (Current directory).

2.1 Súbor so skriptom

Súbor so skriptom zjednodušuje prácu, ak chceme opakovame vyvolať alebo editovať istú postupnosť príkazov. Postupnosť príkazov uložená ako m-súbor sa vyvolá jednoduchým napísaním mena súboru do Command Window.

Príklad 2.1 *Otvorte si súbor* **skript.m**. *Popis súboru:*

1. *%- komentáre. Pridávajú sa do súboru kvôli zvýšeniu prehľadnosti kódu.*
2. *V ďalších dvoch riadkoch sa vytvoria premenné x a y. Poznamenajme, že bodkočiarka na konci riadku spôsobí, že výstup sa nevypíše na obrazovku.*
3. *Vektor x obsahuje* 1000 *prvkov prislúchajúcich rovnomernému deleniu intervalu* $[\frac{\pi}{100}, 10\pi]$.
4. *Vektor y obsahuje hodnoty fukncie sin(x)/x v týchto bodoch. Všimnite si operátor "./". Tento operátor označuje delenie dvoch polí sin(x) a x po zložkách.*
5. *Príkaz* `plot` *nakreslí graf. Viac detailov o grafike sa budeme učiť neskôr.*

Tento súbor môžete spustiť napísaním jeho mena (bez .m) do príkazového okna:

```
>> skript
```

2.2 Súbor s funkciou

Súbor s funkciou je podobný ako súbor so skriptom, avšak má povinnú hlavičku a môže mať vstupné a výstupné argumenty. Medzi súborom so skriptom a súborom s funkciou sú však aj ďalšie, veľmi dôležité rozdiely, o ktorých si povieme trochu neskôr.

Naprogramované „funkcie“ budeme opäť ukladať ako súbry s príponou .m, v každom súbore by pre prehľadnosť mala byť iba jedna funkcia a meno súboru sa musí zhodovať s menom funkcie. Prvý riadok (hlavička) súboru s funkciou musí mať nasledujúcu štruktúru:

```
function [vystupne parametre] = meno_funkcie(vstupne parametre)
```

Vstupné ani výstupné parametre nie sú povinné. Ak funkcia nevracia žiadne výstupné parametre, hranaté zátvorky aj znak -”sa vynechávajú. Meno funkcie musí byť rovnaké ako meno súboru (až na príponu .m): funkciu s vyššie uvedenou hlavičkou by sme teda uložili pod názvom „meno_funkcie.m“. FUnkciu spustíme opäť zabvolaním jej mane z príkazového riadku, spolu so zadaním vstupných parametrov a priradením výstupov do zvolených premenných.

Príklad 2.2 *Vstupnými a výstupnými parametrami môžu byť skaláre, vektory, matice a reťazce. Matlab v podstate nerozlišuje medzi typmi premenných, pokiaľ sa na nich nevykonáva nejaká operácia. Je prípustné, aby vstupným parametrom bol skalár a pri druhom volaní vektor. Na ozrejmenie sa pozrime na nasledujúci príklad.*

Príkazom `y=sin(x)` *vyvoláme zabudovanú funkciu sínus. Ak* `x` *je skalár, potom aj* `y` *bude skalár. Ak* `x` *je vektor, potom aj* `y` *bude vektor. Ak* `x` *je matica, potom aj* `y` *bude matica. Overte tieto tvrdenia jednoduchým experimentom.*

Príklad 2.3 *Otvorte si súbor* **addtwo.m**. *Znakom % začína komentár. Tento riadok nielenže zvyšuje čitateľnosť m-súboru, ale komentár nachádzajúci sa práve na tomto (prvom) mieste v m-súbore sa automaticky stáva súčasťou help-u v Matlabe. Po zavolaní príkazu* `help addtwo` *sa zobrazí práve komentár z tejto pozície. Rovnako sa tento komentár objaví v okne Current Directory pri danom súbore v stĺpci Description. Overte si to!*

Príklad 2.4 *Otvorte súbor* **usporiadaj.m**. *Táto funkcia má jeden vstupný argument, pole reálnych čísel. Na výstupe vracia usporiadané pole. Poznamenajme, že funkcia* `sort` *utriedi prvky poľa v rastúcom poradí.*

Na vyskúšanie funkčnosti tejto funkcie vytvorte najprv pole `a=[pi -10 35 0.15]`. *Potom zavolajte funkciu* `b=usporiadaj(a)`.

2.3 Rozdiely medzi typmi m-súborov

Hlavné a dôležité rozdiely medzi súborom so skriptom a súborom s funkciou sú:

- funkcia má hlavičku so slovíčkom function a zoznamom vstupných a výstupných parametrov. Skript vstupné a výstupné parametre neumožňuje.
- funkcia žije „vo svojom lokálnom svete premenných“, tzn. nepozná žiadne premenné zvonka (pokiaľ niektoré neprinesieme vo forme vstupných parametrov), a podobne, čokoľvek sa vo vnútri funkcie vypočíta, nie je viditeľné ani známe okolitým funkciám alebo príkazovému oknu (pokiaľ niektoré z vypočítaných výsledkov nevynesieme von z funkcie prostredníctvom výstupných parametrov). Na rozdiel od funkcie, skript pozná premenné a ich hodnoty z hlavného programu a tiež všetko, čo sa v skripte vypočíta, je viditeľné hlavnému programu a príkazovému oknu.
- ak tú istú vec naprogramujeme ako skript a ako funkciu, vykonanie funkcie býva obvykle rýchlejšie, čo môže hrať podstatnú rolu pri časovej náročnosti zložitejších úloh.

2.4 Lokálne a globálne premenné

Zadávanie príkazu do Command Line je zhodné s písaním skriptu. Súbor so skriptom nemá začiatok ani koniec, teda všetky skripty na seba nadväzujú a zdieľajú svoje premenné, aj keď nie sú deklarované ako „globálne“. To znamená, že *všetky skripty zdieľajú ten istý workspace*, tzv. „základný pracovný priestor Matlabu“ (angl. Matlab base workspace).

Premenné v súbore s funkciou sú štandardne lokálne. Každá funkcia operuje vo svojom *súkromnom pracovnom priestore*. Funkcie majú prístup len k tým premenným zo súborov so skriptom, ktoré sú buď prinesené ako vstupné argumenty, alebo sú definované ako *globálne*.

2.4.1 Globálne premenné

Ak chcete, aby viacero funkcií zdieľalo tú istú premennú (ale nechcete ju prinášať ako vstupný argument), musíte danú premennú deklarovať ako globálnu vo *všetkých* funkciách. Všetky funkcie používajú ten istý *globálny pracovný priestor* (angl. global workspace). Podobne, ak chcete, aby sa k tejto premennej dalo pristupovať v hlavnom príkazovom okne, musíte ju delarovať ako globálnu aj v ňom alebo v niektorom súbore so skriptom. Deklaráciu `global` musíte urobiť predtým, ako premennú vo funkcii použijete. Na vyčistenie obsahu globálnych premenných použite príkaz `clear global`. Zoznam všetkých globálnych premenných nám vráti príkaz `whos global`.

Upozornenie. Nepoužívajte deklaráciu `global`, kým to nie je nevyhnutné. Vyhnete sa tak viacerým chybám a zmätkom.

Odporúčanie. Ak chcete premennej priradiť nejakú hodnotu a potom ju používať ako globálnu, urobte najprv deklaráciu `global` spolu s priradením hodnoty v príkazovom riadku alebo si vytvorte špeciálny súbor so skriptom, kde si zadefinujete hodnoty všetkých globálnych premenných. Až potom pristúpte k používaniu týchto premenných vo funkciách.

Príklad 2.5 *Otvorte si súbor* **globalne.m** *a súbor* **globalne2.m**. *V súbore globalne.m sa nachádza lokálna premenná y. Keď spustíme tento súbor, premenná y sa neobjaví vo Workspace okne. Navyše táto premenná je iba lokálnou pre danú funkciu, t.j. jej hodnota je neznáma v príkazovom riadku. Rovnako, ak spustíme súbor globalne2.m, kde sa opäť vyskytuje premenná y (ale nie je jej v rámci funkcie globalne2.m priradená žiadna hodnota), Matlab vyhlási chybu. Premenná y v globalne2.m nemá žiadne spojenie s premennou y v globalne.m.*

Pridajme teraz deklaráciu `global y` *do súboru globalne.m. Ak teraz spustíme súbor globalne2.m, premenná y v tomto súbore je ešte stále lokálna. Ak pridáme delaráciu* `global y` *aj do súboru globalne2.m, tak tieto funkcie budú zdieľať tú istú premennú y (všetky funkcie zdieľajú ten istý workspace). Všimnite si, že premenná y je ešte stále nedefinovanou premennou v príkazovom riadku. Ak ju chceme zieľať aj tu, musíme urobiť deklaráciu* `global` *aj priamo v príkazovom riadku.*

2.5 Niekoľko tipov

1. Píšte komentáre. Ak ich nebudete písať, už o niekoľko týždňov si nebudete pamätať, čo ktorá premenná znamená a ako ste to naprogramovali.

2. Čítajte si chybové hlášky. Pomôže vám to pri ladení programu.

3. Pokiaľ je to možné, skúste sa vyhnúť programovaniu cyklov. Zabudované funkcie sa vykonávajú rýchlejšie. Napríklad `y = sin(0:pi/40:pi/2)` je lepšie (rýchlejšie) zadefinovať takto, než pomocou for-cyklu.

2.6 Ďalšie príklady

Príklad 2.6 *Napíšte skript na takúto úlohu: Náhodne vygenerujte celé číslo z množiny* $\{1, 2, ..., 10\}$. *Ak je vygenerované číslo* 1 *alebo* 2, *vypíšte na obrazovku správu: "Pravdepodobnosť je* 20%*",* *v prípade vygenerovania* 3 *alebo* 4 *vypíšte: "Pravdepodobnosťč je* 30%*". V ostatných prípadoch vypíšte: "Pravdepodobnosť je* 50%*".*

Príklad 2.7 *Napíšte funkciu* `s=issymm(A)`, *ktorá na vstupe dostane maticu* `A` *a ako výstup vráti* `s = 1`, *ak je matica symetrická, inak vráti* `s = 0`.

Príklad 2.8 *Napíšte funkciu, ktorej vstupným argumentom bude prirodzené číslo* `n`. *Funkcia vráti štvorcovú maticu rozmerov* $n \times n$, *ktorá má na hlavnej diagonále čísla* $1, 2, ..., n$ *a nuly všade inde.*

Príklad 2.9 *Napíšte funkciu, ktorej vstupným argumentom bude prirodzené číslo* `n`. *Funkcia vráti maticu, ktorej prvky sú* $a_{ij} = 3^{ij}$, *kde* i *je číslo riadku a* j *je číslo stĺpca. Najprv to urobte s použitím niektorého príkazu na kontrolu toku programu (cykly) a potom to skúste bez neho.*

Príklad 2.10 *Napíšte funkciu, ktorá vypíše prvých* n *Fibonnaciho čísel:* $a_1 = 1; a_2 = 1; a_n = a_{n-1} + a_{n-2}$.

Príklad 2.11 *Napíšte funkciu, ktorá dostane na vstupe vektory* `a` *a* `b` *rovnakej dĺžky. Funkcia zostrojí maticu* $C =$

$$\begin{pmatrix} 1 & a_1 & a_2 & \dots & a_n \\ 1 & a_1 + b_1 & a_2 & \dots & a_n \\ 1 & a_1 & a_2 + b_2 & \dots & a_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_1 & a_2 & \dots & a_n + b_n \end{pmatrix}$$

Nájdite determinat tejto matice. Mali by ste spozorovať, že závisí iba od b. *Viete povedať, ako?*