3. Úvod do skriptování

Používání Octave nebo Matlabu jako kalkulačky je sice zajímavé, možnosti těchto programů tím však nejsou ani zdaleka vyčerpány. Jejich síla spočívá v možnosti skriptování, tj. textových souborů obsahujících posloupnosti příkazů a definice funkcí. Velmi jednoduchý skript je uveden v následujícím výpisu.

```
% reseni kvadraticke rovnice
% koeficienty kvadraticke rovnice
a=2;b=-1;c=4;
%diskriminant
D=b^2-4*a*c;
% reseni
x1=(-b+sqrt(D))/2/a;
x2=(-b-sqrt(D))/2/a;
% vypis reseni
x1
x2
```

Pokud uložíme tento skript do souboru `skript1.m`, spustíme jej napsáním jména `skript1` (bez přípony!) do příkazového okna. Co skript dělá, je ukázáno v následujícím, výpisu – z programu Octave.

```
octave:1> skript1
x1 =  0.25000 + 1.39194i
x2 =  0.25000 - 1.39194i
octave:2>
```

Tento výpis uvidíte jen v případě, že skript se nachází v adresáři, který Matlab/Octave prochází při hledání jména skriptu. Jinak se vypíše chybové hlášení, že uvedený příkaz není definován. Pokud se vám chybové hlášení objeví, nevěšte hlavu. Máte dvě možnosti, jak to napravit. Zaprvé, můžeme použít příkaz **cd** (change directory). Máte–li skript uložený v adresáři třeba (v linuxu např. `/home/ja/skripty`, ve Windows `D:\skripty`), napíšete příkaz `cd /home/ja/skripty` (`cd D:\skripty`). Tím se přepnete do daného adresáře, což si můžete ověřit příkazem `pwd` (path to working

directory). Tento příkaz vypíše na obrazovku jméno adresáře, ve kterém se právě nacházíte. Druhý, daleko efektivnější způsob je využití nastavení cesty k adresáři, kde jsou uloženy skripty do seznamu adresářů, které Octave/Matlab automaticky prohledává. V případě Matlabu se přidá adresář se skripty do `PATH` pomocí položky `Set Path` v menu `File`. V Octave se adresář se skripty přidá do prohledávaných adresářů příkazem `addpath(jméno_adresáře)`. Pokud chcete nastavení uchovat, uložte nový seznam adresářů příkazem `savepath`.

3.1 První skript

Zkusme nyní uvedený skript upravit. Budeme chtít zadávat parametry kvadratické rovnice až při běhu skriptu. K tomu se hodí funkce `input(`*'řetězec'*`)`, viz výpis níže:

```
% reseni kvadraticke rovnice
a=input('zadej a: ');
b=input('zadej b: ');
c=input('zadej c: ');
%diskriminant
D=b^2-4*a*c;
% reseni
x1=(-b+sqrt(D))/2/a;
x2=(-b-sqrt(D))/2/a;
% vypis reseni
disp(x1);
disp(x2);
```

Při běhu programu jsme nejdříve vyzváni k zadání konstant *a, b, c*. Kořeny kvadratické rovnice jsou pak vypsány na obrazovku prostřednictvím příkazu `disp(`*proměnná*`)`.

3.2 Funkce jako skript

Mnohem častější používání skriptů jsou funkce a procedury. Ukažme si, jak by vypadala funkce, která řeší kvadratickou rovnici.

```
function [x1,x2]=KvadrRce(a,b,c)
% funkce resi kvadratickou rovnici
% [x1,x2]=KvadrRce(a,b,c),
% kde a,b,c jsou parametry kv.rce
%
  D=b^2-4*a*c; % diskriminant
% reseni
  x1=(-b+sqrt(D))/2/a;
  x2=(-b-sqrt(D))/2/a;
```

Uvedený skript uložíme do souboru pod názvem **KvadrRce.m**. V souboru s koncovkou **m** může být uvedena jen jedna funkce[3] a jméno souboru se **musí** shodovat se jménem funkce. Je třeba dát pozor, neboť Matlab i Octave rozlišují mezi velkými a malými písmeny.

Skript, který obsahuje funkci, začíná klíčovým slovem **function**. Následuje seznam návratových hodnot (v hranatých závorkách) a jméno funkce se vstupními parametry. Řádky začínající znakem % značí komentáře[4]. Je velmi vhodné psát úvodní komentáře, resp. popis co funkce dělá, co jsou vstupní a výstupní hodnoty. Tento komentář (mezi definicí funkce a první příkazem) se zobrazí po zavolání funkce `help`.

```
octave:1> help KvadrRce
funkce resi kvadratickou rovnici
[x1,x2]=KvadrRce(a,b,c),
kde a,b,c jsou parametry kv.rce

KvadrRce is the user-defined function from the file
/home/standa/texty/vyuka/matlab/2/KvadrRce.m
```

Funkce se jednoduše zavolá příkazem `[koren1, koren2]=KvadrRce(1,2,3)`

```
octave:2> [koren1, koren2]=KvadrRce(1,2,3)
koren1 = -1.0000 + 1.4142i
```

[3] Respektive, ve skriptu mohou být deklarovány i další funkce – ty jsou však pouze lokální, tj. mohou je využít jen funkce deklarované ve stejném souboru.

[4] V Octave lze použít i znak #.

```
koren2 = -1.0000 - 1.4142i
octave:3>
```

Do proměnných `koren1` a `koren2` se zapíší hodnoty kořenů rovnice. Není–li příkaz zakončen středníkem, vypíší se výstupní hodnoty na obrazovku.

3.3 Jak psát skripty

Pro psaní skriptů existuje několik doporučení. Shrňme je do následujícího stručného výčtu:
1) jméno skriptu a funkce se MUSÍ shodovat
2) ve skriptu může být jen JEDNA funkce
3) **pište komentáře** !!!!!!!!!
4) pište strukturovaně (zejména cykly, viz dále)

Body 1) a 2) jsou povinné, zbylé dva jsou následování hodná doporučení, která přispívají k přehlednosti.

Funkce zapsaná ve skriptu může volat jiné skripty (funkce). Pokud nechceme, resp. nepotřebujeme definovat nějakou funkci jako skript, můžeme využít tzv. **inline** funkce:

```
octave:1> fx=inline('cos(x)*sin(x)');
octave:2> fx(pi/6)
ans =  0.43301
octave:3> fxy=inline('cos(x)*sin(y)');
octave:4> fxy(pi/6,pi/3)
ans =  0.75000
octave:5>
```

3.4 Cykly a podmínky

Při psaní skriptů se neobejdeme bez cyklů a podmínek. Octave i Matlab disponují cykly `for` a *while* a podmínkami `if` a `switch`. Ukažme si, jak se jednotlivé cykly a podmínky definují a používají. Začněme podmínkou `if`.

3.4.1 Podmínka if

Deklarace podmínky `if` je následující:
`if` (*podmínka*)
příkazy
`else`
příkazy
`end`

Podmínkou může být test na rovnost či nerovnost hodnot proměnných. Matlab umožňuje definici podmínky `není rovno` pomocí zápisu ∼=, Octave umožňuje ještě zápis `!=`. Pro test rovnosti se používají dvě znaménka je rovno (`==`). Znaménko *rovná se* (`=`) je určeno **výhradně** přiřazovacímu příkazu!!! Příkaz `if-then-else` může mít libovolný počet větví `elseif`, což je užitečné při komplikovaných podmínkách.

Zkusme vylepšit náš skript řešící kvadratickou rovnici podmínkou `if` pomocí tak, že budeme testovat vstupní hodnoty:

```
function [x1,x2]=KvadrRce1(a,b,c)
% funkce resi kvadratickou rovnici
% [x1,x2]=KvadrRce(a,b,c),
% kde a,b,c jsou parametry kv.rce
%
D=b^2-4*a*c; % diskriminant
% reseni
if (a~=0) % a je ruzne od nuly
  x1=(-b+sqrt(D))/2/a;
  x2=(-b-sqrt(D))/2/a;
else
  disp(Rovnice neni kvadraticka!!);
  x1=c/b;x2=0.0;
end
```

V případě, že koeficient `a` není roven nule je řešena kvadratická rovnice, pokud je `a==0`, vypíše se zpráva, že rovnice není kvadratická a počítá se kořen `x1`. Do `x2` se dosadí nulová hodnota. To by se správně nemělo, neboť obecně nevíme, zda–li řešením kvadratické rovnice není také hodnota 0.