MATLAB

MATLAB je integrovaným prostředím, s jehož pomocí lze provádět zejména:

- matematické výpočty (snadná a rychlá **práce s maticemi** reálných nebo komplexních čísel),
- modelování,
- analýzu a vizualizaci dat (mnoho hotových funkcí např. pro statistiku),
- měření a zpracování dat,
- vývoj algoritmů,
- návrhy řídicích systémů a další.

Základní komponenty MATLABu:

- výpočetní jádro (maticově orientované),
- grafický subsystém (2D nebo 3D grafy),
- pracovní nástroje (soubor nástrojů umožňující úplné programování aplikací),
- toolboxy (knihovny funkcí rozšiřující možnosti jádra MATLABu; kupují se zvlášť),
- systém Simulink (nadstavba MATLABu pro práci ve schématech a blocích).

Spuštění:

- v prostředí MS Windows: START → Programy → MATLAB...
- v prostředí Linux (Ubuntu): Aplikace → Věda → Matlab (nebo Aplikace → Vzdělávání → Matlab) nebo ALT+F2 a napsat `matlab`

Obr. 1: informační okno při startu MATLABu

Cca od verze 7.0 je pro správné spuštění MATLABu nutné připojení k internetu, resp. správné nastavení licenčního serveru! V některých případech se kvůli licenci otevírá terminálové okno => nezavírejte jej, jinak se MATLAB ukončí!

Ukončení:
File → Exit MATLAB nebo CTRL+Q nebo myší (křížek v pravém horním rohu)

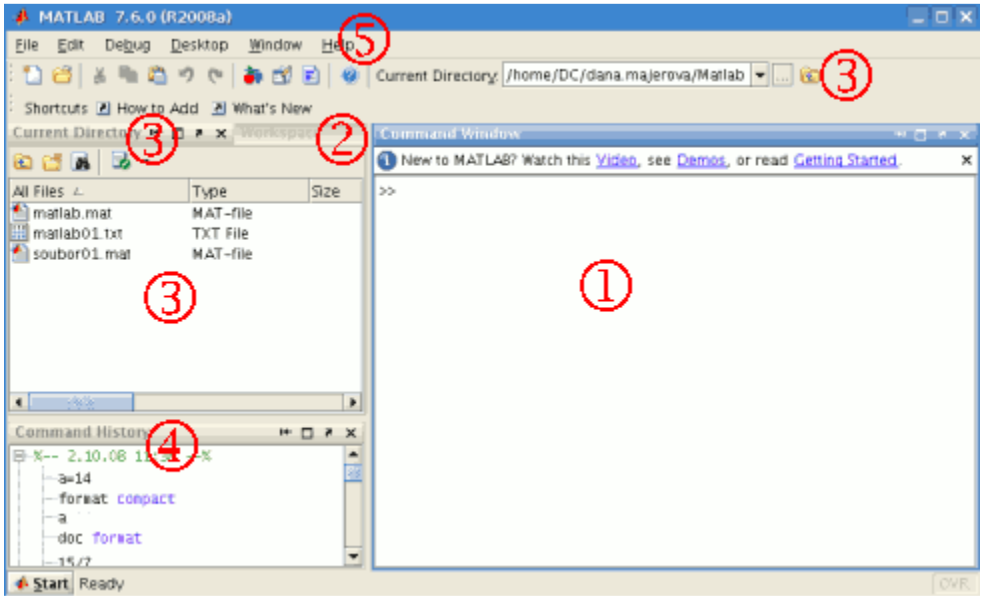
Popis prostředí MATLABu (pracovní plocha)

Po spuštění systému se objeví okno složené z několika částí (obr. 1). Nejdůležitější z nich je (pod)okno Command Window. Uspořádání (pod)oken můžeme změnit, resp. můžeme některá (pod)okna zavřít. Obnovení původního nastavení provádíme pomocí menu View (viz bod 5).

1. Okno **Command Window** - práce v dialogovém režimu
   - zde lze MATLAB používat "jako kalkulačku"
   - jakmile napíšeme a odešleme příkaz, tak je ihned proveden/vyhodnocen (např. `>> 2+3` ENTER)

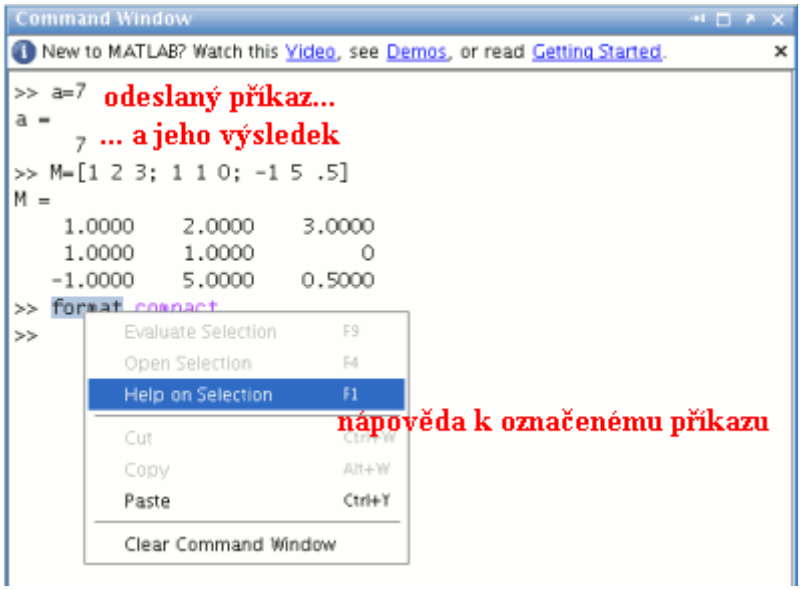
- nepojmenované výsledky se ukládají do proměnné `ans`
- pro editaci příkazů lze používat tyto klávesy:

klávesa	význam
ENTER	odešle řádek ke zpracování
ESC	smaže celý řádek
CTRL+K	smaže zbytek řádku od kurzoru napravo
DEL	smaže jeden znak (za kurzorem)
BACKSPACE	smaže jeden znak (před kurzorem)
HOME	přesun kurzoru na začátek řádku
END	přesun kurzoru na konec řádku
→	posun kurzoru o jeden znak vpravo
←	posun kurzoru o jeden znak vlevo
CTRL+→	posun kurzoru na začátek dalšího slova
CTRL+←	posun kurzoru na začátek předchozího slova
↑ a ↓	listování dříve napsanými příkazy (více v poznámce u bodu 4)
INSERT	vkládání/přepis znaků



Obr. 2: MATLAB 7.6 - pracovní plocha

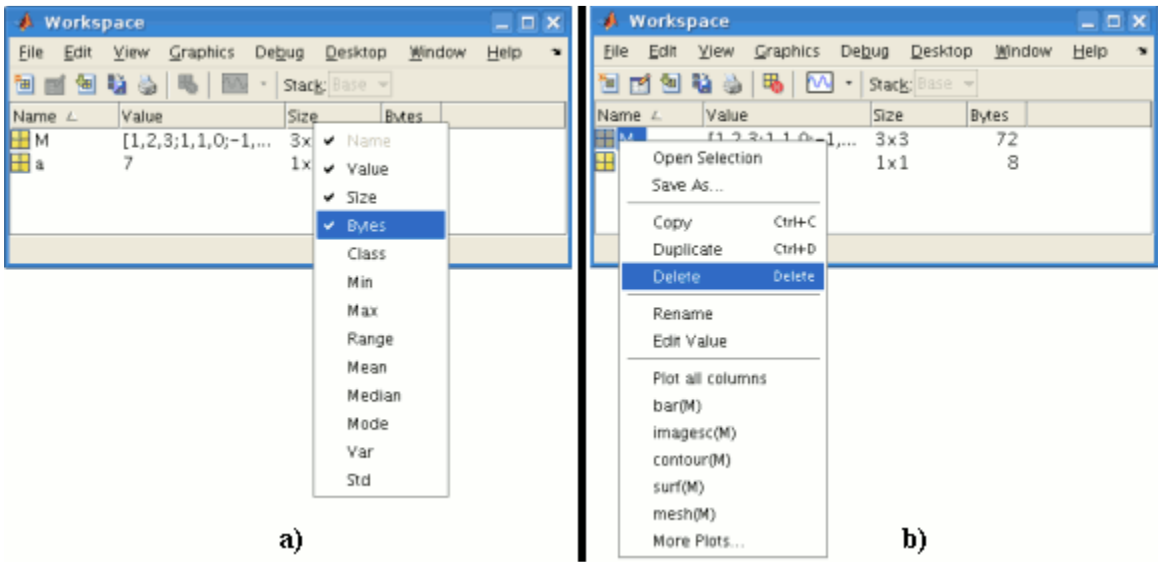
- způsob zobrazování výsledků si lze zvolit pomocí příkazu `format`, například:
  `>> format compact` (kolem výsledků nebudou prázdné řádky)
  `>> format long` (čísla se nebudou vypisovat se čtyřmi, ale se čtrnácti desetinnými místy)
  Původní nastavení příkazu obnovíte příkazem `>> format`. Více viz nápověda (`>> help format`)



Obr. 3: okno Command Window (MATLAB 7.6)

2. Okno **Workspace** - práce s proměnnými

- zobrazuje všechny dostupné proměnné pracovního prostředí
  Pracovní prostředí je základní (Base) nebo lokální (pro funkce - probereme příště při krokování funkcí)
- umožňuje práci s proměnnými, například:
  smazání proměnné: pomocí klávesy DEL,
  zobrazení hodnoty proměnné: dvojklikem,
  uložení všech proměnných do souboru pomocí ikonky s disketou...

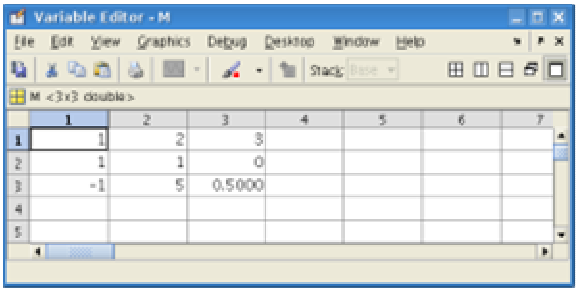


Obr. 4: okno Workspace (MATLAB 7.6) se dvěma proměnnými:
a) úprava sloupců v záhlaví, b) možnosti práce s proměnnou

Poznámky:

- Dvojklikem myši na název proměnné v okně Workspace (ve vyšších verzích MATLABu) se otevře okno pro prohlížení/editaci obsahu proměnné (tzv. Variable Editor nebo Array Editor).

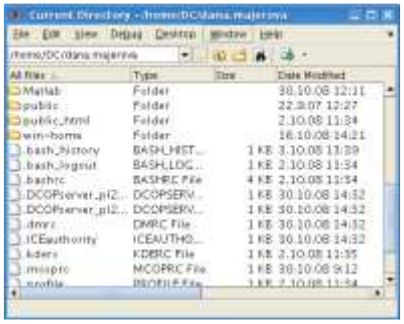
Okno se podobá tabulce v MS Excelu. Příliš rozsáhlé proměnné nebo proměnné složitějšího typu (např. pole buněk) se v něm ale nezobrazí.



Obr. 5: Variable Editor (MATLAB 7.6) s obsahem proměnné (matice *M* řádu 3)

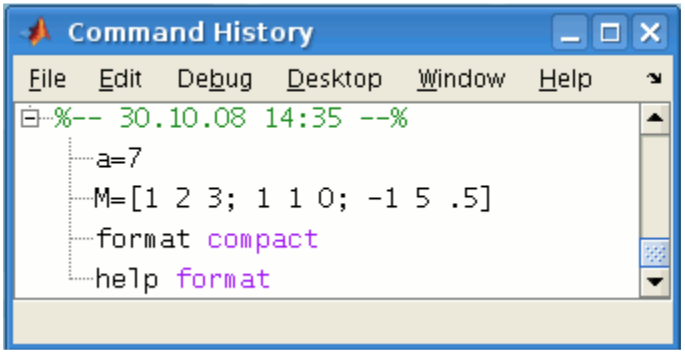
- Okno Workspace nemusíme používat - veškerou práci s proměnnými zvládneme pomocí příkazů (Command Window). Přehled proměnných lze získat příkazem `>> whos` (nebo `>> who` - jen názvy proměnných), výpis proměnné uvidíme po odeslání jejího názvu,... (více viz proměnné).

3. Okno **Current Directory** - pracovní adresář
   - zobrazuje obsah aktuálního (pracovního) adresáře
     Téhož lze docílit pomocí příkazu `>> dir` v Command Window.
   - umožňuje změnit pracovní adresář, např. pomocí ikony se třemi tečkami (pro rychlejší přístup je tato část okna Current Directory obsažena i v panelu nástrojů - viz 3 na obr. 2)
     Téhož lze docílit příkazem
     `>> cd celý_název_adresáře`
     tedy např. `>> cd D:\temp\matlab`
   - kontextové menu souborů umožňuje standardní práci se soubory pracovního adresáře (přejmenování, kopie, přesun, smazání) a navíc také např. otevření M-souboru (viz dále)
     Kontextové menu se otevře po stisku pravého tlačítka myši na názvu daného souboru.
   - soubory lze řadit podle názvu (levé tl. myši na All files), podle typu (levé tl. myši na File Type), podle data poslední změny (levé tl. myši na Last Modified) nebo podle jejich popisu (levé tl. myši na Description)
   - je možné nechat si zobrazit pouze soubory určitého typu - pravé myši na All files (nebo File Type nebo Last Modified nebo na Description) a zaškrtnutím určité volby si vyberete typ zobrazovaných souborů



Obr. 6: okno Current Directory (MATLAB 7.6)

4. Okno **Command History** - přehled použitých příkazů
   - obsahuje všechny použité příkazy. Každé spuštění MATLABu je označeno datem a časem (zelený text mezi procenty)
   - umožňuje opětovné spuštění dříve zadaných příkazů: dvojklik na vybraný příkaz
   - umožňuje úpravu/opravu dříve použitého příkazu: přetažení příkazu myší do Command Window

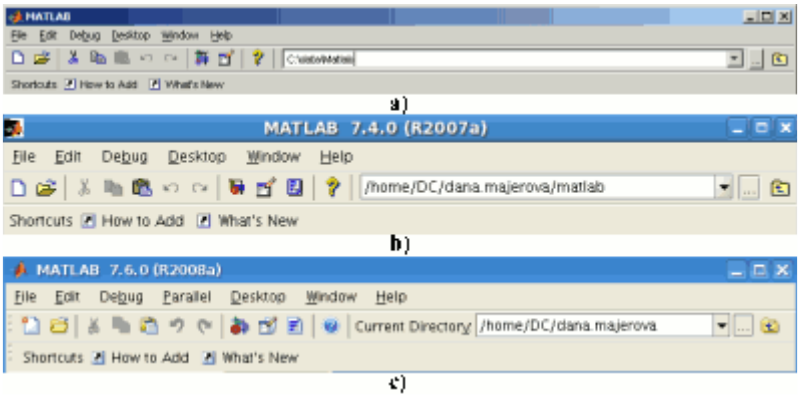


Obr. 7: okno Command History (MATLAB 7.6)

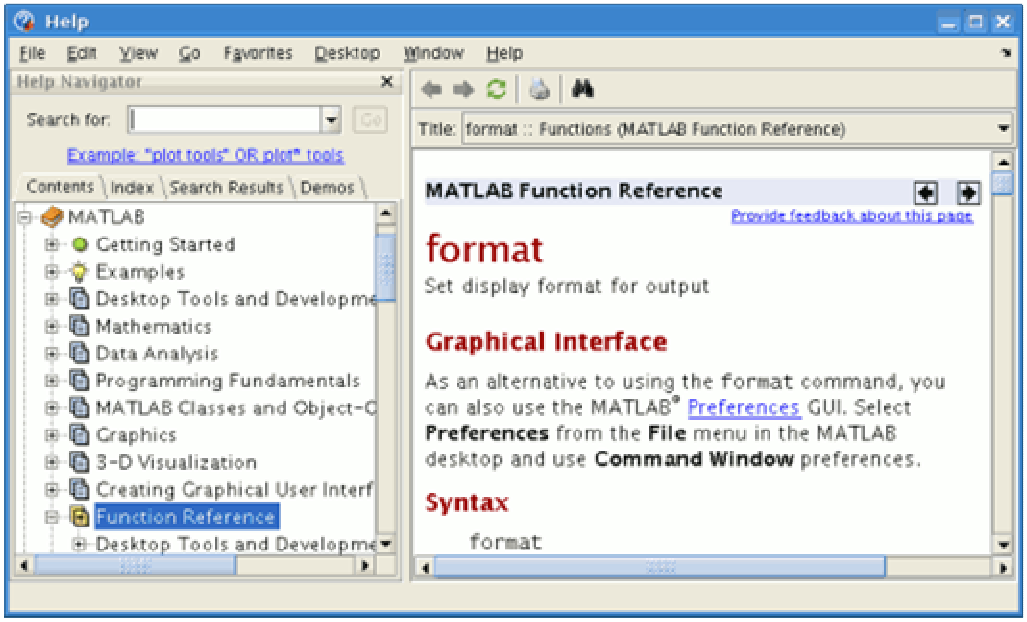
Poznámka: okno Command History nemusíme používat, protože v Command Window lze **listovat použitými příkazy** s použitím šipek (Up, Down). Pokud před stiskem šipky napíšeme začátek hledaného použitého příkazu (alespoň jeden znak), listuje se jen v názvech těch příkazů, které začínají napsaným textem.

5. Menu (nabídky)
   - **File** umožňuje např. vytvořit nebo otevřít M-soubor, ukončit systém,...
   - **Edit** obsahuje např. položky pro kopírování textu přes schránku Windows, ale také umožňuje smazat Command Window, smazat Command History (pak přijdeme o historii příkazů) a smazat Workspace (pak přijdeme o všechny proměnné!)
   - **View** umožňuje nastavit vzhled pracovního prostředí, např. ubrat/přidat výše popsaná okna. !!! Pro nastavení původního vzhledu systému použijte View → Desktop Layout → Default.
   - **Web** umožňuje získat podrobnější informace o výrobci, technické podpoře atd.
   - **Window** je aktivní při práci v grafickém režimu MATLABu.
   - **Help** otevírá rozsáhlou nápovědu k systému MATLAB a jeho součástem.

     Poznámka: nápovědu lze (kromě použití menu Help → MATLAB Help) otevřít také kliknutím na ikonu otazníku v panelu nástrojů. Tím se otevře okno Help Browseru (prohlížeč nápovědy), který sestává ze dvou částí - vlevo je navigátor (umožňuje hledat v rejstříku, vyhledávat podle klíčových slov,...) a vpravo se zobrazuje aktuální téma nápovědy. Všechna témata nápovědy jsou zpracována jako www stránky, takže se v Help Browseru snadno pracuje. Hledání funkcí podle klíčového slova můžete provádět pomocí záložky Search (Hledání), resp. Index (Rejstřík) v samostatném okně nápovědy (obr. 9).



Obr. 8: nabídky (menu): a) MATLAB 7.0, b) MATLAB 7.4, c) MATLAB 7.6



Obr. 9: okno nápovědy - Help Browser (MATLAB 7.6)

Možnosti nápovědy

Příkazy používané v Command Window pro:

A. hledání podle klíčového slova: **`>>lookfor`** (např. `>> lookfor import`)
B. textový výpis nápovědy k funkci (do Command Window): **`>> help`** ***`název_funkce`***, např. `>> help min`
C. interaktivní výpis nápovědy k funkci (do okna nápovědy): **`>> doc`** ***`název_funkce`***, např. `>> doc min`
D. výpis přehledu funkcí z dané skupiny: **`>> help`** ***`skupina`***, např. `>> help datafun`, `>> help ops`,...
E. výpis skupin: **`>> help`**,

Proměnná

Proměnná je objekt, který má svůj název, typ a obsah (hodnotu).

Název proměnné

Název proměnné může obsahovat až 31 znaků. Jsou **povoleny POUZE tyto znaky**: písmena anglické abecedy (a-z, A-Z), číslice (0-9) a podtržítko (_). Číslicí název začínat nesmí.

V názvech jsou rozlišována velká a malá písmena (tzv. vlastnost **case-sensitive**), tedy proměnná `pokus` může existovat současně s proměnnou `Pokus` i třeba `poKuS` - a každá z nich má svou vlastní hodnotu.

Příklady **správných názvů** proměnných: `a`, `A`, `b89x`, `u1`, `pom4`, `pokus_5`, `matice_A`, `fx`.

Příklady **chybných názvů** proměnných (chyba bývá v tom, že se v názvu použije nepovolený znak - v MATLABu mají znaky jako mezera, čárka, tečka, pomlčka, hvězdička atd. speciální význam!):

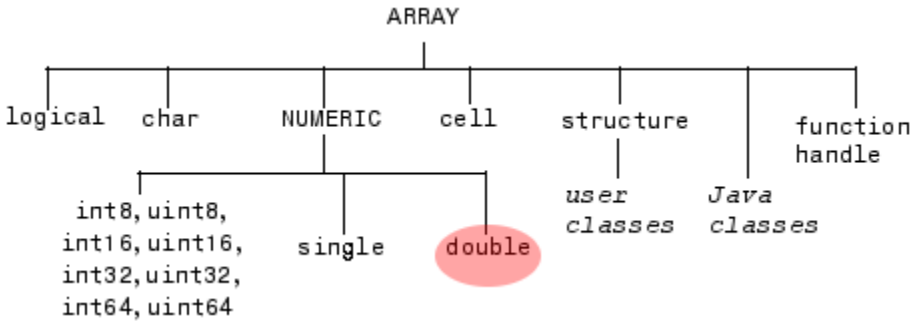
`1pom` ... název nesmí začínat číslicí,
`matice A` ... název nesmí obsahovat mezeru (toto je velmi častá chyba!!!),
`lin.rovnice` ... název nesmí obsahovat tečku (toto je také častá chyba!!!),
`pokus-5` ... název nesmí obsahovat pomlčku (je chápána jako rozdíl proměnných),
`f(x)` ... název nesmí obsahovat závorky.

Typ a hodnota proměnné

Většina proměnných, se kterými budeme pracovat, je datového typu *double* (viz obr. 10). Zde platí základní pravidlo: **KAŽDÁ PROMĚNNÁ JE MATICE komplexních čísel**. Občas se také pracuje s řetězci (*char* = "matice složené ze znaků"), poli buněk (*cell*, podobné maticím, kde každý prvek je jiného typu), strukturami či objekty (*structure*/*user classes*) a symbolickými proměnnými.

Z hlediska rozměrů matic rozlišujeme číselné proměnné na

- matice (*m*x*n*, kde *m*>1, *n*>1)
- vektory (*m*x1 nebo 1x*n*)
- skaláry (1x1, tedy jen jedno číslo)



Obr. 10: datové typy proměnných

Vytvoření proměnné

Pro vytvoření proměnné se používá *přiřazovací příkaz*:
`>> název_proměnné = výraz`.

Výrazy jsou popsány níže. Ukážeme si jednoduché příklady vytvoření proměnných, kdy je použit nejjednodušší výraz - konstanta.

Příklad: **vytvoření skaláru**:
```
>> a=8
>> pom=-2.145
>> skalar8=15e-2
```

- *Desetinná čísla* zadáváme s desetinnou tečkou (ne čárkou!) nebo pomocí zlomku: `8.14` `-0.0002` `13/100`. Pokud je před desetinnou tečkou jenom nula, lze ji vynechat: `.52`
- Čísla lze zadávat také ve *vědeckém formátu* (s použitím písmena e pro označení exponentu 10^e): `1.6e11` `6.122e-8`
- *Imaginární čísla* zadáváme s použitím `i` anebo `j` (viz předdefinované proměnné): `12i` `-6.11j` Např. komplexní číslo 8+0.00032i zadáme jako `8+3.2e-4i`

Vytvoření vektoru nebo matice:

matice typu *m*x*n* vytváříme pomocí **hranatých závorek**, v nichž uvedeme jednotlivé řádky oddělené **středníkem**, přičemž prvky každého řádku (tj. sloupce) oddělujeme **mezerou** nebo **čárkou** (lze použít i obě najednou).
!!! Počty prvků v každém řádku se musejí shodovat, tj. nesmí se stát, že např. některý řádek má méně prvků než ostatní **!!!**
!!! Čárka neslouží jako desetinná čárka (na to je tečka), ale jako oddělovač prvků **!!!**
Chyba vzniklá rozdílným počtem prvků v jednotlivých řádcích (resp. použitím čárky místo desetinné tečky) vypadá takto:
```
??? Error using ==> vertcat
All rows in the bracketed expression must have the same number of columns.
```

Příklad: vytvoření řádkového vektoru:
```
>> v1=[2 0.1 -3.7 4/5 0.14]  nebo
>> u1=[2,0.1, -3.7, 4/5,0.14] % mezery za čárkou nejsou potřeba
```

Příklad: vytvoření sloupcového vektoru:
```
>> v2=[2; 0.1; -3.7; 4/5; 0.14]
```

Příklad: vytvoření matice typu 3x2:
```
>> maticeA=[1 2; 0.1 -3; .7 1/4]
```

Poznámka: přiřazovací příkaz též slouží ke změně hodnoty již existující proměnné, např.
```
>> u1=[1 -3 2].
```

Vytvoření řetězce:

řetězce vytváříme pomocí **apostrofů** (anglických!), v nichž uvedeme text:
```
>> str='Pokusny textik.'
```

Vytvoření pole buněk:

pole buněk vytvoříme pomocí **složených závorek**, v nichž uvedeme jednotlivé hodnoty, např. sloupcové pole buněk se 3 prvky:

```
>> pole1={17; 'Pokus'; [1 2; -1 0]}
```

Zobrazení hodnoty proměnné

Chceme-li zjistit obsah proměnné (= její hodnotu), můžeme použít buď okno Workspace (dvojklik), anebo v dialogovém režimu napíšeme název proměnné:

```
>> název_proměnné
```

Příklad:

```
>> maticeA
>> a
```

Chceme-li zjistit hodnotu prvku matice/vektoru/řetězce, použijeme **kulaté závorky** s indexem uvnitř (více indexů oddělíme **čárkou**):

```
>> maticeA(2,1)
>> v2(3)
>> str(5)
```

Chceme-li zjistit hodnotu prvku v poli buněk, použijeme **složené závorky** s indexem uvnitř:

```
>> pole1{2}
```

Samozřejmě lze kombinovat:

```
>> pole1{3}(2,1)
```

Smazání proměnné

Co s proměnnými, které už nechceme používat? Lze je smazat s použitím okna Workspace, resp. pomocí příkazu `clear`.

`>> clear název_proměnné` - smaže vybranou proměnnou

`>> clear název_proměnné1 název_proměnné2 název_proměnnéN` - smaže vybrané proměnné (jejich názvy od sebe oddělujeme mezerou)

`>> clear` - smaže všechny proměnné z Workspace.

Příklad:

```
>> clear b a  (smaže proměnnou b a proměnnou a)
>> clear  (smaže všechny proměnné)
```

!!! Neuvedeme-li seznam proměnných, smažou se všechny! Smazané proměnné nelze obnovit (pokud nebyly uloženy na disk - viz níže) **!!!**

Předdefinované proměnné

Některé proměnné jsou již definované - patří mezi ně `eps` (malé reálné číslo, 10^{-16}), `i`, `j` (komplexní jednotka) a `pi` (Ludolfovo číslo). Tyto proměnné lze samozřejmě použít i pro uchování jiných hodnot (tím se předdefinovaná hodnota zničí), ale po použití příkazu `>>clear i` (resp. `>>clear`) jsou vráceny do původního stavu.

Příklad:

```
>> i
>> i=7 % nyní zkuste: a=2+i, b=3+5i, c=3+5*i (a,c nejsou komplexní!)
>> clear i
>> i % nyní opět zkuste: a=2+i, b=3+5i, c=3+5*i
```

Proměnná ans

Proměnná `ans` je vytvářena automaticky, pokud některý z příkazů potřebuje vypsat hodnotu, kterou jsme nepřiřadili do žádné proměnné. Proměnná `ans` tedy obsahuje poslední zobrazenou nepojmenovanou hodnotu.
Příklad: pokud proměnná `ans` neexistovala, vytvoří se po vyhodnocení příkazu `>> 2*3-5`. Pokud proměnná `ans` již existovala, tak se změní její hodnota.

Uložení proměnných (na disk)

Někdy se stane, že potřebujeme uchovat nějakou důležitou proměnnou pro použití při příštím spuštění MATLABu (např. musíme přerušit práci). Potřebujeme tedy proměnnou dostat z paměti na disk počítače. K tomu slouží příkaz `save`:

`>> save jméno_souboru název_proměnné` - do zadaného souboru se uloží vybraná proměnná,

`>> save jméno_souboru název_proměnné1 název_proměnné2 název_proměnnéN` - do zadaného souboru se uloží *N* vybraných proměnných (jejich názvy od sebe oddělujeme mezerou),

`>> save jméno_souboru` - do zadaného souboru se uloží všechny proměnné z Workspace,

`>> save název_souboru název_proměnné -ASCII` - do TEXTOVÉHO souboru se uloží vybraná proměnná.

Ve všech případech je vytvořen soubor s příponou MAT (ta je přidána automaticky), který obsahuje názvy a hodnoty vybraných proměnných.

Příklad: uložení dvou proměnných a a `maticeA` do souboru `pokus.mat`:
`>> save pokus a maticeA` (soubor `pokus.mat` vznikne v *pracovním adresáři*)

Poznámka: další možnosti příkazu `save` získáte pomocí `>> help save`.

Import proměnné ze souboru do prostředí MATLABu

Nahrání proměnné (či více proměnných) z MAT-souboru

Je-li na disku nějaký soubor s uloženou jednou nebo více proměnnými (vznikl pomocí `save`), lze tyto proměnné dostat do pracovního prostředí MATLABu pomocí příkazu

`>> load název_souboru`

(u názvu souboru můžeme vynechat jeho příponu MAT).

Příklad:

```
>> clear
>> load pokus
```

Nahrání proměnné z textového souboru (ne MAT-souboru)

Je-li na disku nějaký TEXTOVÝ soubor, kam byla uložena JEDNA proměnná (pomocí příkazu **`>> save název_souboru proměnná -ASCII`**, případně pomocí Notepadu nebo jiného editoru - prvky ve sloupcích oddělené čárkou nebo mezerou nebo tabulátorem, řádky oddělené ENTERem), pak ji příkazem

```
>> load název_souboru
```

dostaneme do Workspace. Oproti výše uvedené možnosti lze takto do Matlabu nahrát pouze JEDNU proměnnou (pokud byl soubor vytvořen příkazem `save` s volbou `-ASCII` pro vícero proměnných, je při pokusu o jeho zpracování příkazem `load` ohlášena chyba - error).

Výhoda: můžete si připravit rozsáhlejší matice nebo vektory např. v Notepadu (každou matici do jiného souboru) a jediným příkazem je nahrát do Matlabu.

Nevýhoda: název proměnné je přebírán z názvu souboru, a proto musí být soubor pojmenován tak, aby neodporoval pravidlům pro vytváření proměnných.

Import dat z MS Excelu (vyberte si jednu z následujících 2 možností)

A. **Využití DDE** (Dynamic Data Exchange) - postup je následující:

1. **Otevřít** příslušný soubor v MS Excelu.
2. Zkontrolovat, zda desetinná čísla jsou psána s desetinnou TEČKOU (čárka je totiž pro MATLAB oddělovačem sloupců v maticích).
   Pokud není zobrazována desetinná tečka, v menu Excelu ji nastavíme jako oddělovač (menu Nástroje→Možnosti, karta Mezinárodní). Není-li v menu Excelu tato možnost, musíme změnit desetinnou čárku na tečku pro všechny aplikace (Start→Nastavení→Ovládací panely, kde na kartě Čísla změníme oddělovač).
3. V MATLABu vytvoříme spojení s Excelem (funkce **`ddeinit`**)
   Např. `>> spojeni=ddeinit('excel','data1.xls')`
   POZOR: pokud je hodnota `spojeni` přesně nula, spojení se nepovedlo (pravděpodobně není otevřen soubor uvedeného názvu).
4. Importujeme data z Excelu do MATLABu (funkce **`ddereq`**)
   Např. `>> x=ddereq(spojeni,'r1c1:r21c1')`, kde 'r1c1:r21c1' znamená, že bereme data od buňky v 1. řádku a 1. sloupci do buňky v 21. řádku a 1. sloupci (tedy od A1 do A21).

B. **Využití funkce `xlsread`**:
   tato funkce načte číselná data ze zadané oblasti XLS souboru. Nečíselné údaje "vně" oblasti budou ignorovány (např. textové záhlaví). Pokud jsou nečíselné údaje i "uvnitř" oblasti, tak se místo nich uloží do MATLABu hodnota `NaN`!
   Plná funkčnost při čtení XLS souborů závisí na tom, zda je možno spustit Excel jako COM server z MATLABu (proto na UNIXových systémech bývá funkčnost omezena na základní - 'basic').

   Příklad:
   - načtení číselných dat ze souboru `test.xls`:

```
>> A = xlsread('testdata2.xls') % namísto nečíselných hodnot se do proměnné A uloží NaN !!!
```

   - načtení číselných dat vybrané oblasti listu 1 ze souboru `test.xls`:

```
>> A = xlsread('testdata2.xls', 1, 'A4:B5')
```

   - načtení číselných dat ze souboru, který se automaticky otevře v MS Excelu, kde vybereme oblast dat:

```
>> A = xlsread('testdata2.xls', -1)
```

Poznámka: pokud nepracujete v prostředí MS Windows (s nainstalovaným MS Excelem) a nefunguje ani funkce `xlsread`, tak zbývá jedině možnost importu dat z textového souboru - viz výše: "Nahrání proměnné (z jiného než MAT-souboru)".

Operátory

Z hlediska rozměrů proměnných rozlišujeme v MATLABu skaláry, vektory a matice. Nyní si uvedeme *operátory pro práci se skaláry* (a postupně si budeme uvádět další; pro nedočkavé studenty: prostudujte si `>> help ops`):

- **aritmetické operátory**
  - unární
    - **unární plus**, např. `+a1` (výsledek je shodný s `a1`)
    - **unární minus**, např. `-a1` (opačné číslo k `a1`code>)
  - binární - vrací výsledek příslušné operace:
    - **sčítání**, např. `a1+a2`
    - **odčítání**, např. `a1-a2`
    - **násobení**, např. `a1*a2`
    - **dělení**, např. `a1/a2`
    - **umocnění**, např. `a1^a2` (konkrétně třeba 3^2 zapíšeme jako `3^2`)
- **relační operátory** - výsledkem je vždy pravda (nenulové číslo, logická 1) nebo nepravda (číslo 0):
  - **menší**, např. `a1<a2`
  - **menší nebo rovno**, např. `a1<=a2`
  - **větší**, např. `a1>a2`
  - **větší nebo rovno**, např. `a1>=a2`
  - **rovnost (je rovno?)**, např. `a1==a2`. POZOR: je velký rozdíl mezi `==` (porovnání) a `=` (přiřazení)!!!
  - **nerovnost (je různé?)**, např. `a1~=a2`
- **logické operátory**
  - unární
    - **negace**, např. `~a1` (změní pravdu na nepravdu, tj. nenulu na nulu, a naopak)
  - binární - výsledek je logická 1 nebo 0; operandy musí být konvertovatelné na 1 nebo 0 (ne tedy např. imaginární čísla)!
    - **logický součin (AND)**, např. `a1&a2`
    - logický součin se zkráceným vyhodnocováním: `a1&&a2` (pokud je `a1` nepravda, tak se `a2` nebude vyhodnocovat vůbec, protože výsledek je vždy logická 0)
    - **logický součet (OR)**, např. `a1|a2`
    - logický součet se zkráceným vyhodnocováním: `a1||a2` (pokud je `a1` pravda, tak se `a2` nebude vyhodnocovat vůbec, protože výsledek je vždy logická 1)

  Poznámka: kromě výše uvedených operátorů nabízí MATLAB ještě funkci `xor(a1,a2)`, která realizuje binární operátor XOR (= exkluzivní OR, negace ekvivalence).

Priorita operátorů

Pořadí, v jakém se budou jednotlivé části výrazu vyhodnocovat, můžeme podle potřeby ovlivnit tím, že použijeme **kulaté závorky** pro ohraničení všech potřebných částí výrazu. Protože každý z výše uvedených operátorů má pevně dané pořadí vyhodnocování, není někdy závorek potřeba. Proto si nyní uvedeme pořadí vyhodnocování (prioritu) operátorů. Operátory v tabulce jsou seřazeny shora dolů podle klesající priority (tj. ty nejnižší se vyhodnocují nejpozději).

	symbol	poznámka
1.	`( )`	*závorky*
2.	`^`	*umocnění*
3.	`+ - ~`	*unární plus, unární minus, negace*
4.	`* /`	*násobení, dělení*
5.	`+ -`	*sčítání, odčítání*
6.	`< <= > >= == ~=`	*relační operátory*
7.	`&`	*logický součin, AND*
8.	`|`	*logický součet, OR*
9.	`&&`	*logický součin se zkráceným vyhodnocením*
10.	`||`	*logický součet se zkráceným vyhodnocením*

Poznámka pro nedočkavé studenty: kompletní přehled priority všech operátorů získáte příkazem `>> help precedence`

Výraz

Výraz je posloupnost konstant, názvů proměnných, operátorů (včetně kulatých závorek - probereme příště) a volání funkcí (probereme příště).

Pokud je výraz smysluplný (MATLABem vyhodnotitelný), tak po jeho napsání a stisku klávesy ENTER je výraz *ihned vyhodnocen*. Vyhodnocením výrazu vzniká vždy nějaká *hodnota*.

Výslednou hodnotu výrazu můžeme odebrat (uložit do nějaké proměnné - viz *přiřazovací příkaz*), jinak je uložena do proměnné `ans` (viz výše) a zobrazena, například:

```
>> 3+2*0.5
ans =
        4
```

Pokud nepotřebujeme vypočtenou hodnotu vidět, lze její zobrazení potlačit - za výrazem napíšeme **středník**. Středník tedy slouží (kromě oddělení řádků matice) také k **potlačení výpisu výsledku** výrazu.

Obvykle se při potlačeném výpisu výsledku používá přiřazení hodnoty výrazu do nějaké proměnné, protože jinak vypočtená hodnota zanikne. Proměnná vytvořená přiřazením s potlačeným výpisem samozřejmě vznikne ve Workspace. Typicky tedy píšeme:

```
>> proměnná=výraz;
```

Příklad:

výraz	odebrání hodnoty?	zobrazení výsledku?	vznik (změna) ans?
`>> b = 5*a+3`	ano	ano	ne
`>> 5*a+3`	ne	ano	ano
`>> b = 5*a+3;`	ano	ne	ne
`>> 5*a+3;`	ne	ne	ne [ano ve vyšších verzích]

Poznámka: poslední případ je ve starších verzích MATLABu (cca do verze 6.5?) k ničemu, protože výraz se sice vyhodnotí, ale jeho výsledek se nikdo nedozví.

Příkaz

Pokud napíšeme a odešleme nějaký výraz ke zpracování, MATLAB ho bere v podstatě jako *příkaz* k nějaké činnosti (vyhodnocení výrazu). Kromě odeslání výrazů však existuje spousta "čistokrevných" příkazů, např. příkaz přiřazovací. Základní přehled:

- přiřazovací příkaz (`=`), jehož syntaxe je
  ```
  >> název_proměnné = výraz
  ```
- podmíněný příkaz, větvení (`if`) - viz dále
- přepínač (`switch`)
- cyklus neboli smyčka (`for`, `while`)

Dále MATLAB obsahuje příkazy (většinou jde o funkce s voláním podobným příkazu) pro:

- práci s adresářem (např. `cd`, `dir`)
- práci s proměnnými (např. `save`, `load`, `clear`)
- formátování výpisu proměnných v Command Window (`format`)
- ...

Jeden příkaz na více řádků

Někdy se stane, že potřebujeme napsat jeden příkaz na více řádků. K tomu účelu použijeme **tři tečky**. Tři tečky se zvýrazňují modře a znamenají, že MATLAB má počkat s provedením příkazu, protože ještě není celý. Pokud MATLAB čeká na dokončení příkazu, tak příkazový řádek nezačíná `>>`.

Příklad:

```
>> A = [1.5 -3 4.1; 2...
1.14 5 0.2; 7 15.1 8/31 0];
```

nebo

```
>> A = [1.5 -3 4.1; 2...
1.14 5 0.2; 7 15.1...
8/31 0];
```

Nevýhoda: příkaz se ukládá po částech i do Command History, takže se v případě opětovného použití musí volat zase po částech.

Potlačení výpisu výsledku

Některé příkazy vypisují své výsledky (např. `>> x = 215`). Pokud nepotřebujeme tyto výsledky vidět, lze **potlačit výpis výsledků příkazu** (stejně jako u výrazů) napsáním **středník**u na konec příkazu (`>> x = 215;`).

Násilné ukončení příkazu (CTRL+C)

Příkaz, který se právě provádí, násilím ukončíme pomocí stisku **CTRL+C**. Většinou se CTRL+C používá, pokud jsme "vyrobili" nekonečný cyklus nebo jsme zapomněli na středník a necháváme vypisovat příliš velké matice.

Více příkazů na jednom řádku

Doposud jsme příkazy ukončovali vždy pomocí klávesy ENTER, která zároveň odeslala příkaz ke zpracování. Občas ale chceme odeslat ke zpracování více příkazů najednou. Tehdy můžeme použít buď M-soubory, anebo zápis více příkazů na jeden řádek, přičemž k odělení jednotlivých příkazů se používá čárka nebo středník (čárka jen odděluje příkazy, středník navíc potlačuje výpis jejich výsledků).
Příklad:

```
>> m=8, n=3; vysledek=3*m+0.5*n
```

Komentáře (poznámky)

Komentáře slouží většinou k vysvětlení významu jednotlivých příkazů, skriptů nebo funkcí. Je velmi vhodné je používat (ať už kvůli vlastní skleróze nebo kvůli kolegům, se kterými možná budete výsledky své práce sdílet). Komentáře u uživatelských funkcí navíc slouží jako nápověda, kterou umí MATLAB zobrazovat např. příkazem `>> help` *`název_fce`*

Komenář začíná znakem `%` (procento) a končí spolu s koncem řádku. Text komentáře bývá označen zeleně (pokud jste si nezvolili jinou barvu).

Všechny komentáře jsou MATLABem ignorovány, tj. nejsou vyhodnocovány.

Příklad:

```
>> x = [125/100 0.9-9]; % vytvoreni vektoru x s hodnotami 1.25 a -8.1, bez vypisu
```

Podmíněný příkaz (větvení)

Základní syntaxe:

```
if podmínka
   příkazy
end
```

Popis:

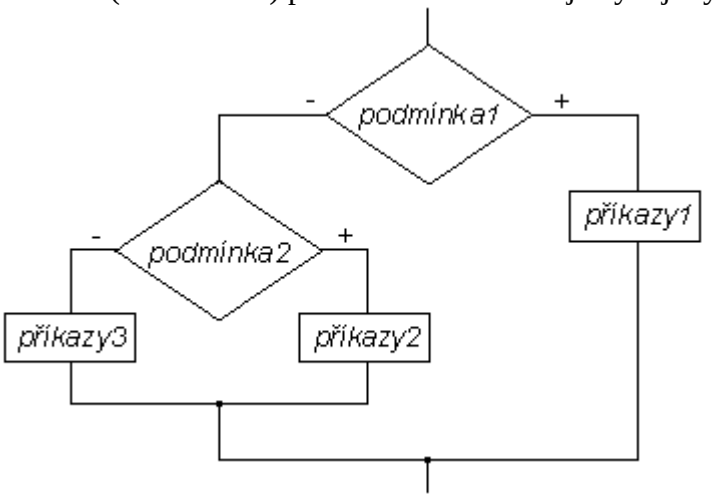
- `if` - klíčové slovo, označuje začátek příkazu (musí být uvedeno na začátku, patří k němu `end`)

Rozšířená syntaxe:

```
if podmínka1
   příkazy1
elseif podmínka2
   příkazy2
else
   příkazy3
end
```

- *podmínka* - libovolný výraz s logickou hodnotou 0 nebo 1
- *příkazy* - lib. příkazy, které se mají provádět v případě splnění příslušné podmínky
- `elseif` - klíčové slovo, které označuje další větev s podmínkou - nahrazuje `if` vnořený do `else` (smí se vyskytnout 0-*N*krát)
- `else` - klíčové slovo, které označuje větev "jinak" (smí se vyskytnout nanejvýš jednou)
- `end` - klíčové slovo, které označuje konec celého příkazu.

Činnost (rozšířeného) příkazu `if` demonstruje vývojový diagram:



Poznámky:

- V podmínkách se většinou používají relační nebo binární logické operátory.
- V případě, že součástí podmínky je matice typu *m*x*n*, je podmínka pravdivá jen tehdy, pokud je pravdivá pro všechny prvky této matice.
- V případě, že v nějaké podmínce použijeme binární logický operátor `&&` nebo `||` a po vyhodnocení prvního argumentu je už zřejmý výsledek (kdy se to může stát?), tak se druhý argument nevyhodnocuje!!!
- Jednotlivé příkazy `if` se mohou vnořovat, tj. uvnitř `if`u může být další příkaz `if`.

Činnost příkazu `if` obecně: nejprve se testuje pravdivost *podmínky1*. Pokud platí, provedou se *příkazy1* a ostatní větve příkazu jsou ignorovány. Pokud *podmínka1* neplatila, začne se testovat *podmínka2* - když platí, provedou se *příkazy2* a zbylé větve jsou ignorovány; neplatí-li, pokračuje se další větví `elseif` (když je ještě nějaká)... V případě, že ani jedna z podmínek neplatila, jsou provedeny příkazy ve větvi `else` (je-li přítomna, jinak není proveden žádný příkaz uvnitř `if`u).

Příklad 1 - podle hodnoty věku (jehož zadání si vyžádá funkce `input`), vypíšeme hlášení (pomocí funkce `disp` pro výpis řetězců):

```
>> c = input('Zadej vek: '); if c<20, disp('jsi nas'), elseif c>30,... % více řádků
disp('starochu'), else, disp('jeste by to slo'), end
```

a po stisknutí ENTERu jsme hned dotázáni na věk (napíšeme `21`) a po stisku ENTER uvidíme ihned výsledek:

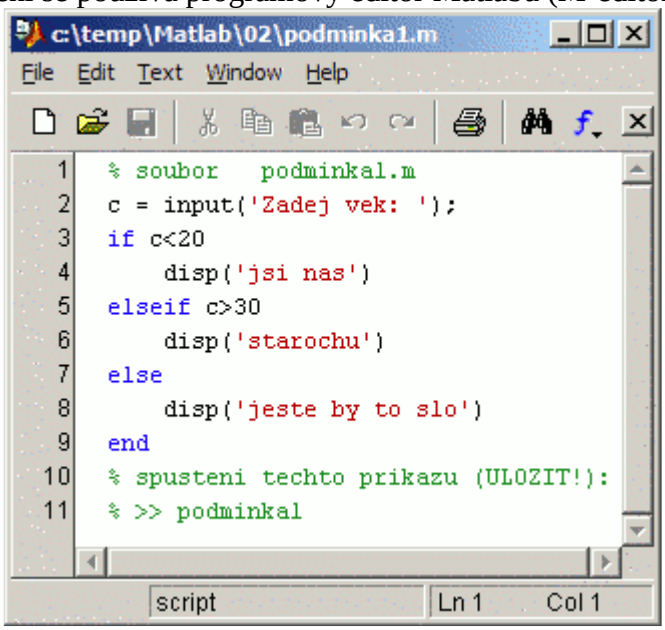
```
Zadej vek: 21
jeste by to slo
>>
```

Všimněte si:
- po zadání části příkazu `if` MATLAB čeká na jeho dokončení (nevypisují se znaky >>, i kdyby nebylo použito `...`),
- funkce `disp` vytiskne řetězec hned od začátku řádku (nejsou před ním mezery).

Nevýhoda:
- pokud bychom chtěli vyzkoušet stejný příkaz pro jiný věk, musíme všechno napsat/zkopírovat znovu (tj. práce v Command Window se v případě složených příkazů stává neefektivní),
- do Command History se ukládá i číslo, které zadáváme funkci `input` - v rámci historie příkazů je to zbytečný údaj,
- pokud uděláme při psaní složeného příkazu chybu (např. zapomeneme oddělit příkazy čárkou nebo středníkem), tak vše musíme napsat/zkopírovat znovu!

Řešení:
Složené **příkazy uložit do souboru**, k jehož vytvoření se používá programový editor Matlabu (M-editor):
- je dobré si PŘEDEM nastavit **pracovní adresář**, např. `>> cd C:\temp\Matlab`
- menu File → New → M-file otevře M-editor s prázdným souborem
- napíšeme příkazy (M-editor sám odsazuje), k výpisu řetězců použijeme funkci `disp`
- ULOŽÍME pod vhodným názvem (musí splňovat pravidla pro název proměnných - více v části "M-soubory"), např. `podminka1.m`
- spustíme z Command Window pomocí `>> podminka1` (bez přípony!)

Právě jsme vytvořili a spustili náš první skript - více v části "M-soubory".



Přepínač (switch)

Syntaxe:

```
switch výraz
   case hodnota1
      příkaz1
   case hodnota2
      příkaz2
   ...
   otherwise
```

Popis:

- `switch` - klíčové slovo, označuje začátek příkazu (musí být uvedeno na začátku, patří k němu `end`)
- *výraz* - testovaný výraz, většinou název nějaké proměnné. Smí být jenom typu skalár nebo řetězec!

```
        příkazN
end
```

- *příkazy* - lib. příkazy, které se mají provádět v případě, že výraz nabývá dané hodnoty
- `case` - klíčové slovo, které označuje větev, kde se provede porovnání výrazu s uvedenou hodnotou (musí se vyskytnout alespoň jednou)
- `otherwise` - klíčové slovo, které označuje větev "jinak" (smí se vyskytnout nanejvýš jednou)
- `end` - klíčové slovo, které označuje konec celého příkazu.

Poznámky:

- Příkaz postupně porovnává hodnotu `výrazu` s každou hodnotou uvnitř `case` (pro skaláry se používá `==` a pro řetězce funkce `strcmp`) - při první nalezené shodě se provedou odpovídající `příkazy` a přepínač SKONČÍ.
- Oproti jazyku C **je provedena nanejvýš jedna větev** `case`, a není tedy potřeba uvádět `break`.
- Je-li přítomna větev `otherwise`, pak je provedena právě jedna větev (buď ta odpovídající `case`, anebo `otherwise`).
- V případě, kdy chceme stejné příkazy provést pro několik různých hodnot, uvedeme všechny tyto hodnoty jako jednořádkové pole buněk u některého `case`, tedy:

```
case {hodnota1, hodnota2, hodnota3}
   příkazy
```

- Příkazy jedné větve lze psát i do jednoho řádku, ale musíme je od `hodnoty` oddělit čárkou/středníkem, tedy:

```
case hodnota, příkaz1, příkaz2, příkaz3
```

Příklad 2 - podle hodnoty jména (jehož zadání si vyžádá funkce `input`) vypíšeme pozdrav:

```
% soubor vetveni.m
jm = input('zadej jmeno: ');
switch(jm)                          % nebo jen:  switch jm
    case 'Jan', disp('Ahoj Honzo');
    case {'Karel','Sheafraidh'}     % více hodnot
        disp('Cauky kamo');
    case 'noname'
        disp('zdar nikdo');
    otherwise
        disp('ahoj taky');
end
% spusteni >> vetveni
```

Spuštění (pokud je soubor `vetveni.m` uložen v akt. adresáři):

```
>> vetveni % ENTER
zadej jmeno: 'Petr' % ENTER Pozor: musíte zadat ŘETĚZEC (tj. apostrofy okolo)!
ahoj taky
```

Nevýhoda: v Command History zbytečně zůstává řetězec (jméno, které jsme zadali)

Řešení:
Z uvedeného souboru uděláme nový (`vetveni2.m`) tak, aby uvnitř byla definována **funkce** (podrobnosti příště) s jedním vstupem:

```
% funkce v souboru vetveni2.m
function vetveni2(jm)                  % název fce musí být shodný se jménem souboru,
switch(jm)
    case 'Jan', disp('Ahoj Honzo');
    case {'Karel','Sheafraidh'}
        disp('Cauky kamo');
    case 'noname'
        disp('zdar nikdo');
    otherwise
        disp('ahoj taky');
end
% spusteni >> vetveni2('Jan')
```

!!! Funkce se musí jmenovat stejně jako soubor (bez přípony)!!!

Spuštění:

```
>> vetveni2('Petr') % ENTER
ahoj taky
```

Cykly

Cyklus slouží pro zápis příkazů, které mají být prováděny opakovaně (několikrát za sebou). Počet opakování těchto příkazů závisí na nějaké podmínce nebo může být předem známý. V MATLABu existují dva základní typy cyklů:

- cyklus *s předem známým počtem opakování* (zvaný též iterační cyklus nebo cyklus s výčtem hodnot; v MATLABu realizovaný pomocí `for`)
- cyklus *řízený podmínkou* (zvaný též indukční cyklus; v MATLABu realizovaný pomocí `while`)

Příkazy uvnitř cyklu se nazývají *tělo cyklu*.

Cyklus s předem známým počtem opakování (iterační cyklus) - `for`

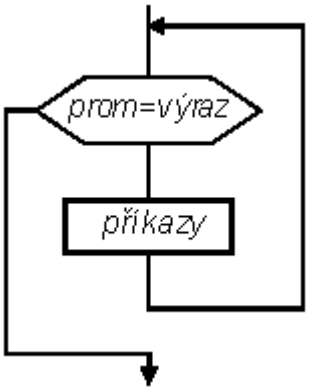
Syntaxe:

```
for výraz=proměnná
   příkazy
end
```

Popis:

- `for` - klíčové slovo, označuje začátek příkazu (musí být uvedeno na začátku, patří k němu `end`)
- *`proměnná`* - platný název proměnné, tzv. *řídicí proměnná cyklu*. V každé iteraci (tj. v každém průchodu cyklem) nabude jedné hodnoty *`výraz`*u
- *`výraz`* - libovolný vektor; jeho délka (velikost) udává počet opakování cyklu - většinou je to vektor generovaný operátorem dvojtečka
  Je-li *`výraz`*em matice, tak bude řídicí proměnná obsahovat postupně každý její sloupec (a počet sloupců udává celkový počet opakování)
- *`příkazy`* - lib. příkazy, které se mají provádět opakovaně (tolikrát, kolik prvků má *`výraz`*)
- `end` - klíčové slovo, které označuje konec celého příkazu.

Činnost (průběh provádění cyklu) `for`



Na začátku se zjistí délka (*n*) výrazu a pokud je nenulová, *n*-krát se provede tělo cyklu (příkazy). V každé iteraci je řídicí proměnná cyklu rovna jednomu prvku vektoru (začíná se prvním prvkem a pokračuje se po řadě). Po vyčerpání všech hodnot vektoru cyklus skončí, tzn. pokračuje se prováděním příkazů za jeho koncem (který je označen klíčovým slovem `end`), pokud tam nějaké jsou.

Konec cyklu může nastat předčasně, pokud je v těle cyklu obsažen příkaz `break` (viz dále) nebo `return` (probereme příště).

Časté použití: pro tvorbu matic (jejichž prvky jsou generovány podle nějakého vzorce) a dále při zpracovávání matice prvek po prvku.

Poznámky:

- Cyklus `for` může být vnořen do libovolného jiného složeného příkazu (např. do `if`u nebo do jiného cyklu).
- Mezi příkazy těla cyklu můžeme napsat jakýkoli jiný složený příkaz, např. `if` nebo další cyklus.

Příklad 3 - 10x vypíšeme `ahoj` (soubor `ahoj10.m`):

```
% soubor ahoj10.m
for i=1:10
    disp('ahoj');
end
% >> ahoj10
```

Spuštění v Command Window (soubor musí být ULOŽEN v pracovním adresáři jako `ahoj10.m`):

```
>> ahoj10 % ENTER
ahoj
ahoj
ahoj
ahoj
ahoj
ahoj
ahoj
ahoj
ahoj
ahoj
```

Vylepšení: počet výpisů pozdravu by byl vstupem funkce s názvem `ahoj` (soubor `ahoj.m`):

```
% soubor ahoj.m ... Nkrat
function ahoj(pocet)
for i=1:pocet
    % disp('ahoj');                ... původní výpis
    disp(['ahoj ' num2str(i)]);   % výpis i s pořadovým číslem
end
% >> ahoj(25)
```

Spuštění v Command Window (soubor musí být ULOŽEN v pracovním adresáři jako `ahoj.m`):

```
>> ahoj(3) % ENTER
ahoj 1
ahoj 2
ahoj 3
```

Vyzkoušejte také: `>> ahoj(-5)`, `>> ahoj(0)`, `>> ahoj(25)`, `>> ahoj(1.75)` - PROČ fungují právě takto?

Cyklus řízený podmínkou - `while`

Syntaxe:

```
while podmínka
   příkazy
end
```

Popis:

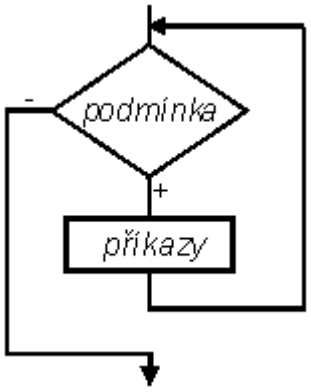
- `while` - klíčové slovo, označuje začátek příkazu (musí být uvedeno na začátku, patří k němu `end`)
- *`podmínka`* - tzv. řídicí podmínka cyklu. Je to lib. výraz, který vrací (skalární) hodnotu 1 nebo 0 (tj. pravda/nepravda). Většinou se v podmínkách používají relační nebo binární logické operátory
- *`příkazy`* - lib. příkazy, které se mají provádět, pokud platí *`podmínka`*
- `end` - klíčové slovo, které označuje konec celého příkazu.

Poznámky:

- V případě, že v nějaké podmínce použijeme binární logický operátor `&&` nebo `||` a po vyhodnocení prvního argumentu je už zřejmý výsledek (kdy se to může stát?), tak se druhý argument nevyhodnocuje (tj. využije se zkrácené vyhodnocování).
- Cyklus `while` může být vnořen do libovolného jiného složeného příkazu (např. do `if`u nebo do jiného cyklu).
- Mezi příkazy těla cyklu můžeme napsat jakýkoli jiný složený příkaz, např. `if` nebo další cyklus.

Činnost (průběh provádění cyklu) `while`

Činnost cyklu lze vyjádřit větou: *dokud platí podmínka, prováděj příkazy*.
Na začátku se testuje platnost řídicí podmínky (tj. je vyhodnocena). Pokud *podmínka platí* (tj. výraz má nenulovou hodnotu), provedou se všechny příkazy těla cyklu. Poté se ZNOVU testuje podmínka a pokud platí, provedou se znovu všechny příkazy těla cyklu atd. atd., dokud se při testování podmínky nezjistí, že *podmínka neplatí* (tj. hodnota výrazu je nulová). Tehdy cyklus skončí, tzn. pokračuje se prováděním příkazů za jeho koncem (který je označen klíčovým slovem `end`), pokud tam nějaké jsou.



Konec cyklu může nastat i po prvním testu podmínky (je-li nulová), takže se může stát, že příkazy v těle cyklu se neprovedou ani jednou.

Konec cyklu může nastat předčasně, pokud je v těle cyklu obsažen příkaz `break` (viz dále) nebo `return` (probereme příště).

Protože se podmínka testuje opakovaně a závisí na ní ukončení cyklu, měli bychom dodržovat následující **doporučení**:
řídicí podmínka cyklu by měla být ovlivňována prováděním příkazů v těle cyklu tak, aby jednou přestala platit. Jinak totiž cyklus nikdy neskončí (tzv. *nekonečný cyklus*) - poznáme to tím, že v Command Window se neobjeví >> a zpravidla se také neustále něco vypisuje. Pokud chceme ukončit provádění nekonečného cyklu, stiskneme CTRL+C.

Použití: cyklus `while` používáme vždy, když potřebujeme opakovat nějakou činnost závislou na pravdivosti dané podmínky, ale předem neznáme počet těchto opakování. Tato podmínka se nazývá *řídící podmínka cyklu*.

> **Příklad 4** - nekonečný cyklus:
>
> ```
> >> while 1, disp('jedu...'), end
> ```
>
> Podmínka je vždy pravdivá (má totiž pořád hodnotu 1), a proto se bude vypisovat `jedu...`, dokud nestisknete CTRL+C.

Break

Příkaz **`break`** se používá k ukončení cyklu (`for` nebo `while`).

Pokud se jedná o několik vnořených cyklů, tak `break` ukončí ten nejvnitřnější cyklus.

Příkaz `break` nelze použít jinde než uvnitř cyklů.

Continue

Příkaz **`continue`** ukončí aktuální iteraci a začne provádět další (pokud jsou splněny podmínky pro pokračování cyklu, tj. je-li ještě nějaká hodnota *`výraz`*u u `for`, resp. je-li pravdivá podmínka u `while`).

Pokud se jedná o několik vnořených cyklů, tak `continue` ukončí iteraci toho nejvnitřnějšího cyklu.

Pozn.: ve starších verzích Matlabu (před 6.5?) nebyl tento příkaz k dispozici.

MATLAB - režimy práce

Dialogový režim

- je přístupný v okně Command Window
- v tomto režimu je možno Matlab používat jako "inteligentní kalkulačku s funkcemi"

- napsané výrazy/příkazy se po odeslání (stiskem klávesy ENTER) ihned vyhodnocují/vykonávají
- umožňuje práci s proměnnými pracovního prostředí (Workspace)

Programový režim

- slouží pro editaci M-souborů (skriptů nebo uživatelských funkcí)
- je spjat se samostatným oknem M-editoru (= editor M-souborů, kde je barevně zvýrazňována syntaxe příkazů)
- obsahuje debugger, který umožňuje hledat chyby ve funkcích/skriptech

Grafický režim

- slouží pro vizualizaci výsledků
- otevírá samostatná okna s názvem Figure

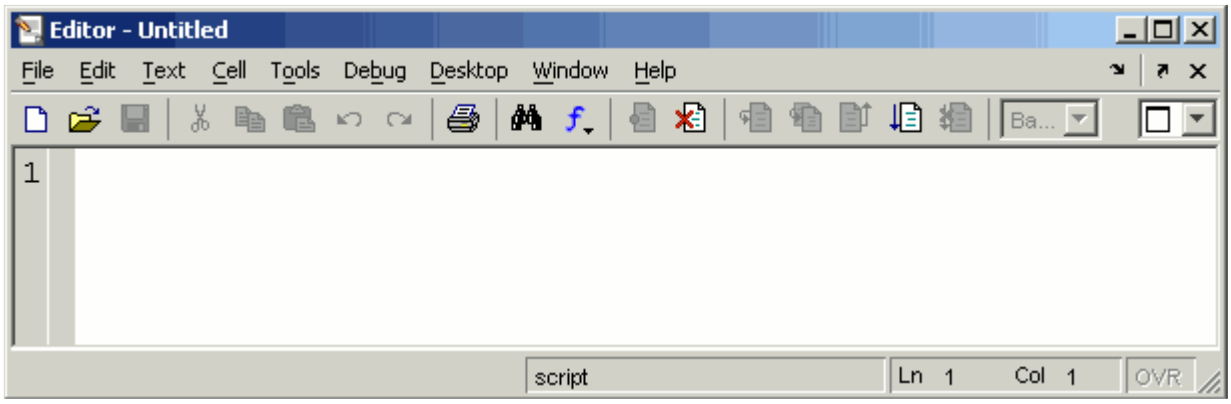
M-soubory

M-soubory slouží k ukládání posloupností příkazů (skripty) nebo k ukládání uživatelských funkcí (funkce).

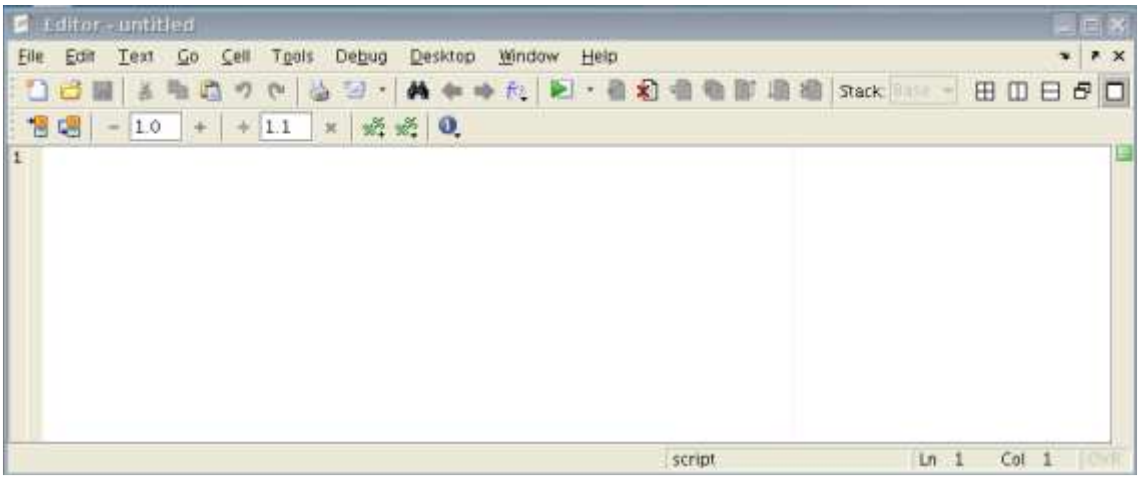
Poznámka: M-soubory jsou obyčejné textové soubory, a proto je lze psát i v libovolném textovém editoru. Z důvodu vyššího komfortu (zvýraznění syntaxe, možnost krokování) je však součástí MATLABu také M-editor/Debugger, který se otevírá v samostatném okně po otevření nebo vytvoření M-souboru (skriptu či funkce).

M-editor

M-editor/Debugger je spuštěn v samostatném okně po otevření (File→Open) nebo vytvoření (File→New →M-file) M-souboru. Slouží k pohodlné editaci M-souborů. Navíc umožňuje krokovat obsah M-souborů (tj. kontrolovat provádění jejich jednotlivých příkazů) - ukážeme si příště.



Obr. 11a: prázdné okno M-editoru (MATLAB 6.5)



Obr. 11b: prázdné okno M-editoru (MATLAB 7.6)

Skripty

Skript je posloupnost příkazů uložených do souboru. Každý skript *pracuje s proměnnými pracovního prostředí* (Workspace), takže může vytvářet nové nebo mazat či měnit vybrané. Výsledky skriptu tedy zůstávají v pracovním prostředí i po jeho skončení. Skripty samozřejmě mohou volat jiné skripty nebo funkce, vytvářet grafická okna, vypisovat do Command Window,...

> *Poznámka*: V době psaní skriptu se jeho příkazy samozřejmě neprovádějí - k tomu potřebu;jeme spustit skript v Command Window (viz Spuštění skriptu).

Vytvoření skriptu

1. Nejprve si v MATLABu **nastavíme pracovní adresář** (nejlépe na lokálním disku).
2. **Vytvoření nového nebo otevření existujícího skriptu**
   M-soubor, který bude obsahovat skript, **vytvoříme** například pomocí menu File→New→M-file, čímž se také otevře okno M-editoru/Debuggeru. Pokud chceme opravit již existující skript, musíme jej **otevřít** (například pomocí menu File→Open).
3. **Zápis skriptu**
   Do prázdného M-souboru zapíšeme všechny příkazy, které má skript provést - příkazy píšeme stejně jako v Command Window, jen s tím rozdílem, že se po napsání neprovádějí. Takto vytvoříme **kód skriptu** (posloupnost příkazů).
4. **Uložení skriptu**
   Máme-li napsaný kód skriptu, musíme celý M-soubor uložit pod nějakým jménem na disk (do pracovního adresáře). **Uložení** provedeme pomocí menu File→Save, kde zkontrolujeme pracovní adresář a zadáme jméno skriptu:
   jméno skriptu musí splňovat **stejná pravidla jako název proměnné**, přestože se jedná o jméno souboru. Je to z toho důvodu, aby MATLAB mohl skript spustit (viz níže). Jméno M-souboru se skriptem tedy může obsahovat POUZE písmena anglické abecedy, podtržítko a číslice (číslicí nesmí začínat)**!!!**

Spuštění skriptu

Spuštění skriptu dává MATLABu pokyn k vykonání jeho příkazů. Před spuštěním musí být **skript uložen**! Skript můžeme spustit buď

a. v Command Window - stačí zadat jméno M-souboru bez přípony (`>> jméno`, tj. skript `graf1.m` spustíme příkazem `>> graf1`), anebo
b. v M-editoru/Debuggeru pomocí menu Debug→Run (klávesa F5).

Příklad 1 - vytvoříme skript `parabola.m`, který kreslí graf funkce x^2:

parabola.m

```
% GRAF1 - vykresleni paraboly
x = -3:0.1:3; % vektor hodnot z intervalu [-3;3] s krokem 0,1
y = x.^2; % zavisle promenna
plot(x,y) % graf (parabola)
```

...uložíme jej a spustíme příkazem

```
>> parabola
```

...pokud je vše v pořádku, objeví se okno s grafem. V opačném případě musíme najít a opravit chybu, uložit soubor a znovu jej spustit.

Zobrazení obsahu skriptu v Command Window

Obsah skriptu (M-souboru) vidíme v M-editoru/Debuggeru, ale můžeme jej zobrazit také v Command Window příkazem `>> type název`, kde *název* je jméno skriptu (název M-souboru bez přípony).