# POČÍTAČOVÁ PODPORA V ELEKTROTECHNICE

Ing. Lenka Šroubová, Ph.D.
lsroubov@kte.zcu.cz

Ing. Petr Kropík, Ph.D.
pkropik@kte.zcu.cz

Katedra teoretické elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Místnost: EK602


FACULTY OF ELECTRICAL ENGINEERING
UNIVERSITY OF WEST BOHEMIA

# Řízení běhu výpočtu

Příklad: Součet náhodných čísel ve vektoru s počtem prvků, které zadá uživatel, pomocí `sum()` a pomocí cyklu `for`.
Ověříme, že příliš výpisů na obrazovku zpomaluje tím, že zobrazíme okno (proužek) o stavu výpočtu.

`h = waitbar(x,'zpráva')` – vytváří a zobrazuje proužek částečně zaplněný podle velikosti $x$, které by mělo být mezi 0 a 1. V horní části se zobrazuje aktuální text zprávy.

*Pozn.*:
`rem(x,y)` – zbytek po celočíselném dělení, např.

```
rem(3,2)
ans =     1
rem(4,2)
ans =     0
rem(78,5)
ans =     3
```

3/2=1, zbytek 1 (3 je liché číslo)

4/2=2, zbytek 0 (4 je sudé číslo)

78/5=15, zbytek 3

```
function testsumacewbar
while(1)
 pocet_cisel = input('Zadej pocet cisel vetsi nez milion: ');
 if(pocet_cisel>1e6)
     break
 end
end
vektor = rand(1,pocet_cisel);
tic
prvni = sum(vektor);
toc
disp('Pomoci sum: ');
disp(prvni)
```

```
tic
druhy = 0;
h = waitbar(0,'Makam jako drak...');
for cykl=1:length(vektor)
    druhy = druhy + vektor(cykl);
    if(rem(cykl,10000) ~= 0)
        continue;
    end
    waitbar(cykl/length(vektor));
end
close(h);
toc
disp('Pomoci cyklu for: ');
disp(druhy)
end
```

```
function testsumacewbar
while(1)
 pocet_cisel = input('Zadej pocet cisel vetsi nez milion: ');
 if(pocet_cisel>1e6)
     break
 end
end
vektor = rand(1,pocet_cisel);
tic
prvni = sum(vektor);
toc
disp('Pomoci sum: ');
disp(prvni)
```

musí být zadán počet větší než $10^6$

součet prvků ve vektoru pomocí sum

```
tic
druhy = 0;
h = waitbar(0,'Makam jako drak...');
for cykl=1:length(vektor)
    druhy = druhy + vektor(cykl);
    if(rem(cykl,10000) ~= 0)
        continue;
    end
    waitbar(cykl/length(vektor));
end
close(h);
toc
disp('Pomoci cyklu for: ');
disp(druhy)
end
```

```
function testsumacewbar
while(1)
 pocet_cisel = input('Zadej pocet cisel vetsi nez milion: ');
 if(pocet_cisel>1e6)
     break
 end
end
vektor = rand(1,pocet_cisel);
tic
prvni = sum(vektor);
toc
disp('Pomoci sum: ');
disp(prvni)
```

musí být zadán počet větší než $10^6$

jen každý 10000cí obrat provede výpis

součet prvků ve vektoru pomocí sum

```
tic
druhy = 0;
h = waitbar(0,'Makam jako drak...');
for cykl=1:length(vektor)
    druhy = druhy + vektor(cykl);
    if(rem(cykl,10000) ~= 0)
        continue;
    end
    waitbar(cykl/length(vektor));
end
close(h);
toc
disp('Pomoci cyklu for: ');
disp(druhy)
end
```

zbytek po dělení

zavření okna

součet prvků ve vektoru pomocí for

# Řízení běhu výpočtu

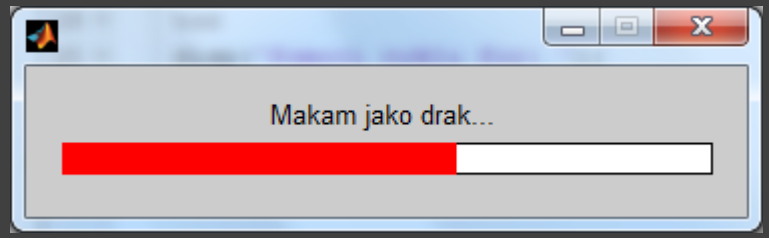
Pokračování příkladu:
Volání funkce jejím názvem, např.:

```
testsumacewbar
Zadej pocet cisel vetsi nez milion: 5
Zadej pocet cisel vetsi nez milion: 50
Zadej pocet cisel vetsi nez milion: 1e7
Elapsed time is 0.011707 seconds.
Pomoci sum:
   4.9997e+006

Elapsed time is 2.582634 seconds.
Pomoci cyklu for:
   4.9997e+006
```

Spotřebovaný čas pomocí sum

Spotřebovaný čas pomocí for s výpisem



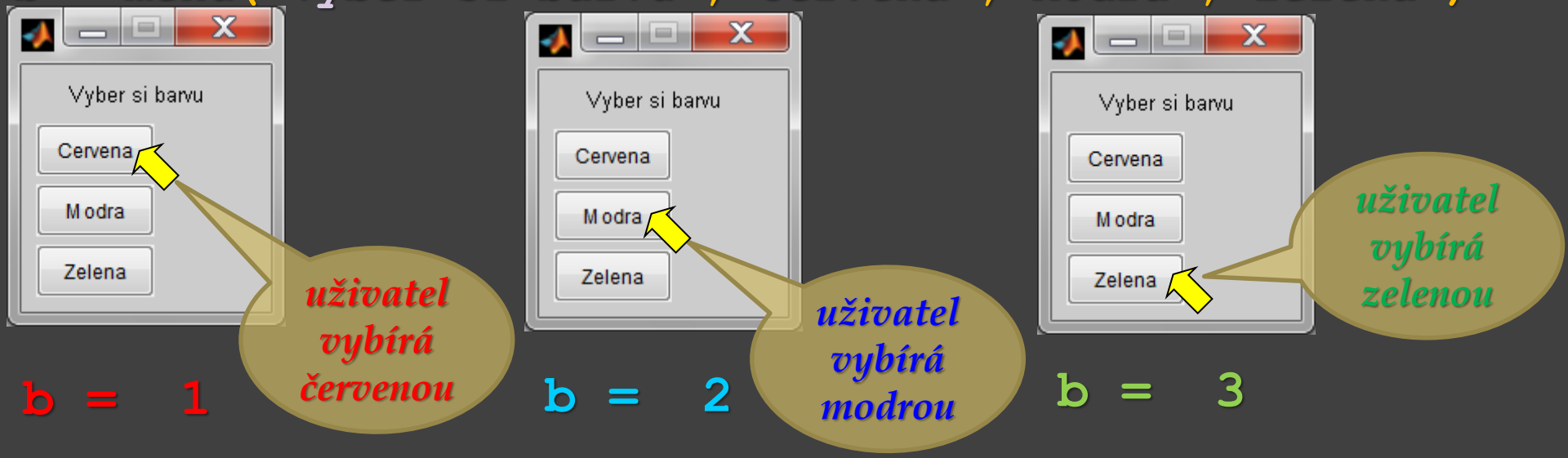
# Funkce pro vstup a výstup

*Pozn.*: ke **vstupu** dat

Mimo **`input`** lze použít i **`menu`**. Tento příkaz obsahuje kromě výzvy pro uživatele i **několik možných voleb** (počet volíme podle potřeby).

Uživatel si zvolí volbu, jejíž pořadové číslo bude pak přiřazeno proměnné. Např.:

```
b = menu('Vyber si barvu','Cervena','Modra','Zelena')
```




Stejný výsledek lze dosáhnout i příkazem **`input`** (a např. **`if`**, příp. **`switch`**). Funkce **`menu`** celý postup pouze může usnadnit.

# Funkce pro vstup a výstup

Příklad: Vytvoření uživatelské funkce **vyberZmenu** bez parametrů. Tato funkce **vyberZmenu** bude vykreslovat grafy křivek dané rovnicemi:

$f(x) = \mathbf{sin}^3(x^2)$ červeně,
$f(x) = \mathbf{2} \cdot \mathbf{sin}^3(x^2)$ modře,
$f(x) = \mathbf{3} \cdot \mathbf{sin}^3(x^2)$ zeleně.

Křivka bude vykreslena podle volby uživatele vždy pro $x$ z intervalu od **0** do **5**.

Pro výpočet $y = \mathbf{sin}^3(x^2)$ bude vytvořena další funkce **sin3x2** s jedním vstupním parametrem $x$ a jedním výstupním parametrem $y$. Funkce **sin3x2** bude volána ve funkci **vyberZmenu** a její výstup bude využit při vykreslení grafů.

*Pozn.*:
`pause` – čekání na stisk klávesy
`pause(1.45)` – čeká 1,45 sec. a pak pokračuje

```
function vyberZmenu
while(1)
 volba = menu('Vyber si barvu','Cervena','Modra','Zelena','Konec');
    switch volba
        case 1
            barva = 'Cervena';
            plot(0:0.01:5,sin3x2(0:0.01:5),'r');
            title([barva,' sin^3x^2 '])
        case 2
            barva = 'Modra';
            plot(0:0.01:5,2.*sin3x2(0:0.01:5),'b');
            title([barva,' 2.sin^3x^2 '])
        case 3
            barva = 'Zelena';
            plot(0:0.01:5,3.*sin3x2(0:0.01:5),'g');
            title([barva,' 3.sin^3x^2 '])
        case 4
            disp('Konec, okna budou zavřena')
            pause(1.5); close all; % zavře všechna grafická okna
            break;
        otherwise
            break;
    end
end
end
```

```
function y = sin3x2(x)
y=(sin(x.^2)).^3;
end
```

```
function vyberZmenu
while(1)
 volba = menu('Vyber si barvu','Cervena','Modra','Zelena','Konec');
    switch volba
        case 1
            barva = 'Cervena';
            plot(0:0.01:5,sin3x2(0:0.01:5),'r');
            title([barva,' sin^3x^2 '])
        case 2
            barva = 'Modra';
            plot(0:0.01:5,2.*sin3x2(0:0.01:5),'b');
            title([barva,' 2.sin^3x^2 '])
        case 3
            barva = 'Zelena';
            plot(0:0.01:5,3.*sin3x2(0:0.01:5),'g');
            title([barva,' 3.sin^3x^2 '])
        case 4
            disp('Konec, okna budou zavřena')
            pause(1.5); close all; % zavře všechna grafická okna
            break;
        otherwise
            break;
    end
end
end
```

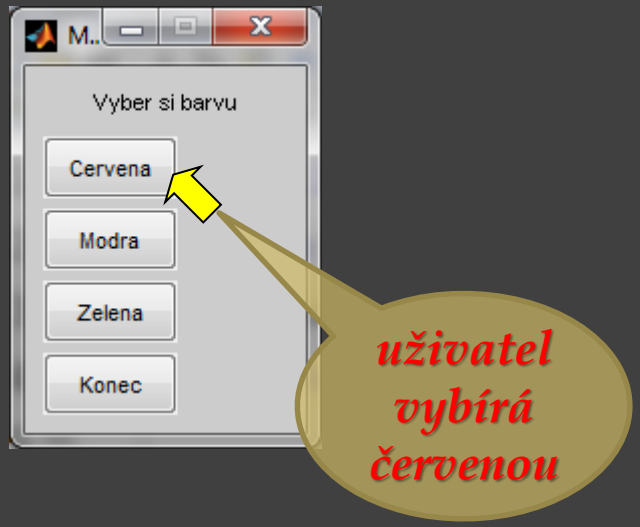
1 2 3 4

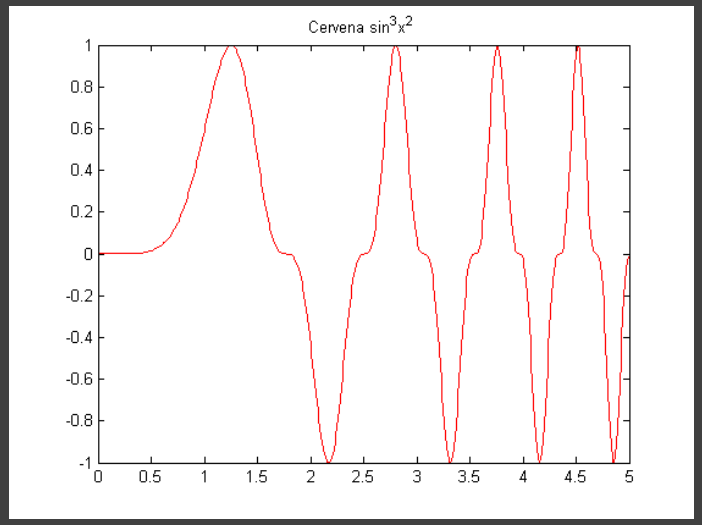
Toto by nemělo nastat

```
function y = sin3x2(x)
y=(sin(x.^2)).^3;
end
```

# Funkce pro vstup a výstup

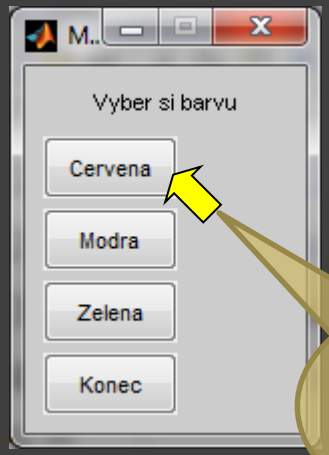
Pokračování příkladu: Volání funkce **`vyberZmenu`**



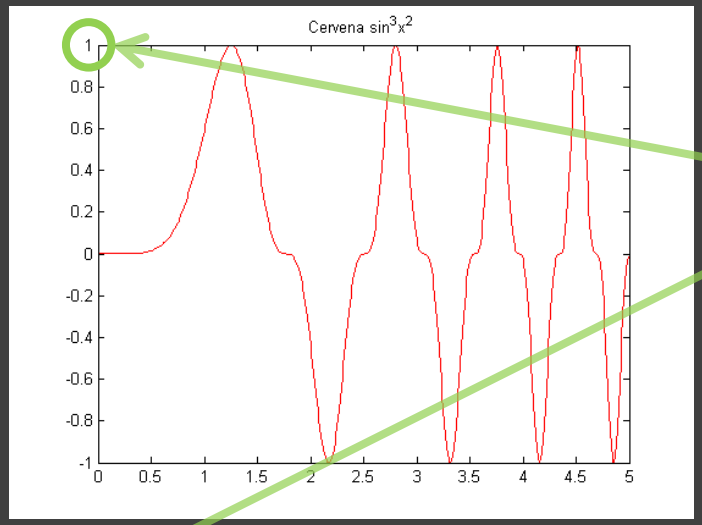



# Funkce pro vstup a výstup

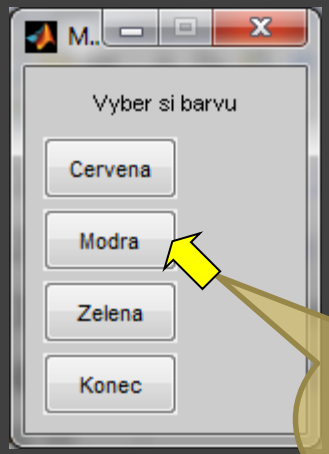
Pokračování příkladu: Volání funkce **`vyberZmenu`**



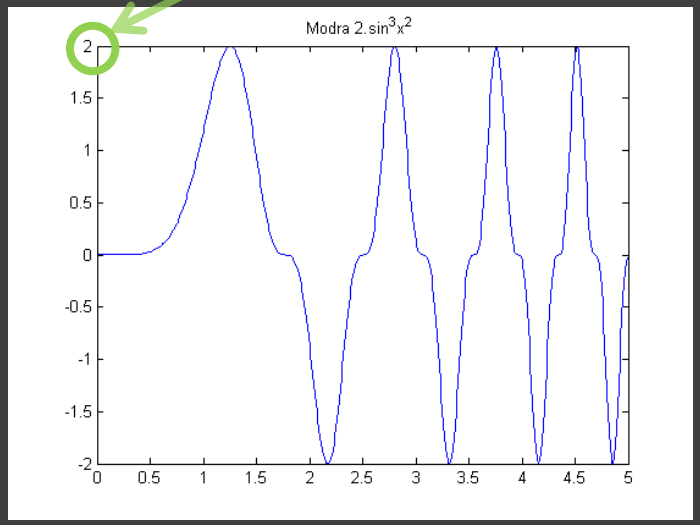

*uživatel vybírá červenou*




**Rozsahy os *y* se liší**

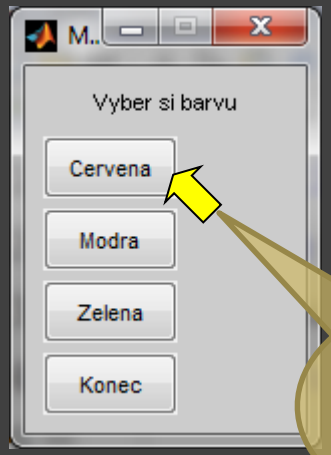



*uživatel vybírá modrou*

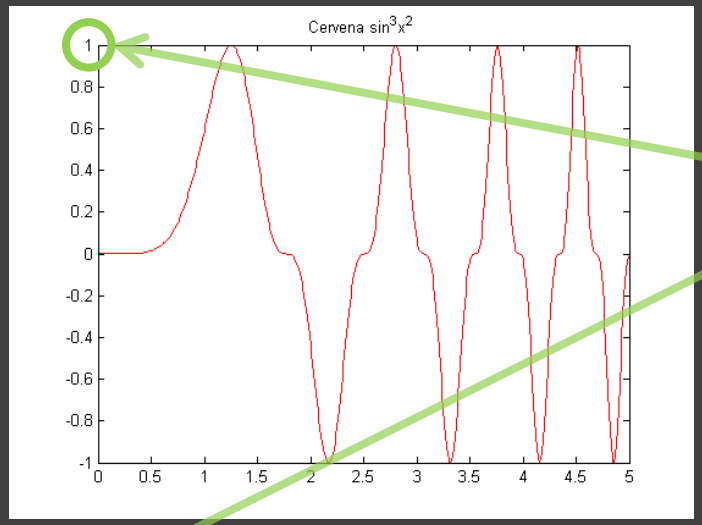


# Funkce pro vstup a výstup

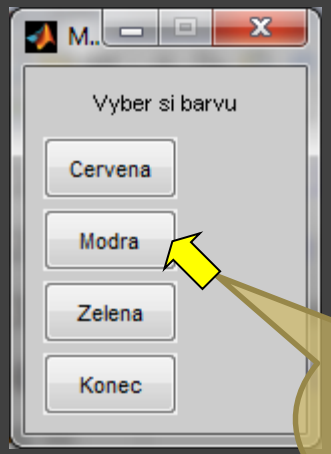
Pokračování příkladu: Volání funkce **`vyberZmenu`**



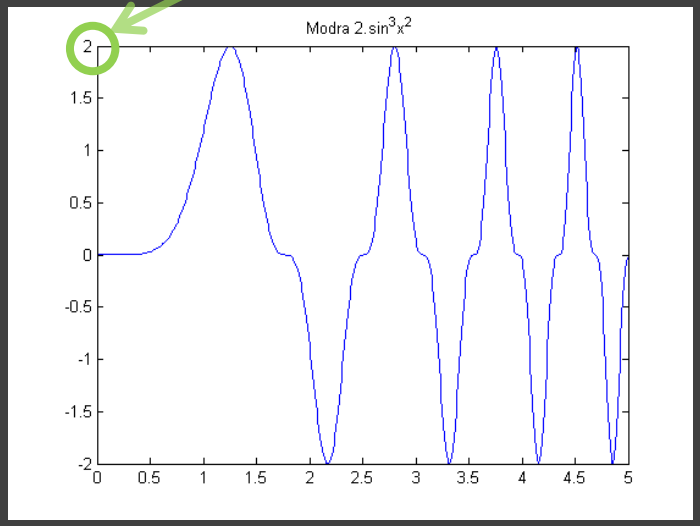

*uživatel vybírá červenou*

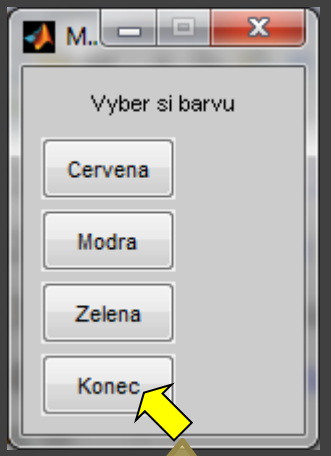



**Rozsahy os *y* se liší**




*uživatel vybírá modrou*




*uživatel vybírá konec*

**`Konec, okna budou zavřena`**

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**`fprintf ('formatovací sekvence', argumenty)`**

– vypíše argumenty v požadovaném formátu v pořadí v jakém jsou zapsány. Formátovací sekvence – začínají znakem `%` :

`%d` nebo `%i` – desítkové číslo (znaménkové)
`%o` – číslo v osmičkové soustavě
`%u` – desítkové číslo (neznaménkové)
`%x` nebo `%X` – číslo v šestnáctkové soustavě (`a` – `f` nebo `A` – `F`)
`%f` – desetinné číslo
`%e` nebo `%E` – desetinné číslo v exponenciálním tvaru (`e` nebo `E`)
`%g` nebo `%G` – použito `%f` nebo `%e` resp. `%E` – exponenciální tvar se použije, je-li třeba, tj. je-li číslo moc velké nebo malé. Navíc vypouští nevýznamné nuly
`%c` – tisk znaku (z proměnné)
`%s` – tisk textového řetězce (z proměnné)

*Pozn.*
`%%` - tiskne znak `%`

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**`fprintf`**

Důležité řídící znaky:

`\n` – nová řádka
`\t` – tabelátor
`\r` – návrat na začátek téhož řádku (záleží na operačním systému)
`\a` – v některých operačních systémech – pípnutí
`\b` – Backspace – vymazání předchozího znaku
`\\` - tiskne znak \ (zpětné lomítko)

Příklad:

```
a = 56; b = 134;
```

– tisk hodnot uložených v proměnných `a`, `b` na obrazovku

```
fprintf('%d %d\n', a, b);
56 134
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

Příklad:

```
a = 56; b = 134;
```

- tisk hodnot na 8 znaků, zleva doplněny **mezery**

```
fprintf('%8d %8d\n',a,b);
      56      134
```

- tisk hodnot na 8 znaků, zleva doplní poprvé **nuly**, podruhé **mezery**

```
fprintf('%08d %8d\n',a,b);
00000056      134
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

Příklad:

```
a = 56; b = 134;
```

- tisk hodnot na 8 znaků, zleva doplněny **mezery**

```
fprintf('%8d %8d\n',a,b);
      56      134
```

- tisk hodnot na 8 znaků, zleva doplní poprvé **nuly**, podruhé **mezery**

```
fprintf('%08d %8d\n',a,b);
00000056      134
```

6+2=8

5+3=8

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

– uvedené číslo říká "alespoň 8 znaků", nesmí oříznout delší celé číslo.

Příklad:

```
a = 56; b = 134;
```

- tisk hodnot na 8 znaků, zleva doplněny **mezery**

```
fprintf('%8d %8d\n',a,b);
      56      134
```

- tisk hodnot na 8 znaků, zleva doplní poprvé **nuly**, podruhé **mezery**

```
fprintf('%08d %8d\n',a,b);
00000056      134
```

6+2=8

5+3=8

```
fprintf('%6d\n',1000);
  1000
fprintf('%4d\n',1000);
1000
fprintf('%2d\n',1000);
1000
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

– uvedené číslo říká "alespoň 8 znaků", nesmí oříznout delší celé číslo.

Příklad:

```
a = 56; b = 134;
```

- tisk hodnot na 8 znaků, zleva doplněny mezery

```
fprintf('%8d %8d\n',a,b);
      56      134
```

číslo 1000 má 4 číslice, 6 znaků => 2 mezery, 2 +4= 6

- tisk hodnot na 8 znaků, zleva doplní poprvé nuly, podruhé mezery

```
fprintf('%08d %8d\n',a,b);
00000056      134
```

```
fprintf('%6d\n',1000);
  1000
fprintf('%4d\n',1000);
1000
fprintf('%2d\n',1000);
1000
```

6+2=8

5+3=8

4 znaky => žádná mezera, 0 +4= 4

2 znaky => žádná mezera, celé číslo 1000

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**`fprintf`**

Příklad:

```
x = 123.3456;
```

– výpis **desetinného** čísla

```
fprintf('%f\n', x);
123.345600
```

– výpis **desetinného** čísla na **tři desetinná místa**

```
fprintf('%.3f\n', x);
123.346
```

`fprintf` zaokrouhluje

– výpis **desetinného** čísla na **dvě desetinná místa**

```
fprintf('%.2f\n', x);
123.35
```

– výpis **desetinného** čísla na **osm desetinných míst**

```
fprintf('%.8f\n', x);
123.34560000
```

*Pozn.* `%f` – zobrazení čísla v pevné řádové čárce

# Funkce pro vstup a výstup

## Formátovaný textový výstup

`fprintf`

Příklad:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (tj. na celou část zbývá minimálně **12**, **celá část** má **3** číslice =>doplněno **9 mezerami**)

```
fprintf('%15.2f\n', x);
           123.35
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

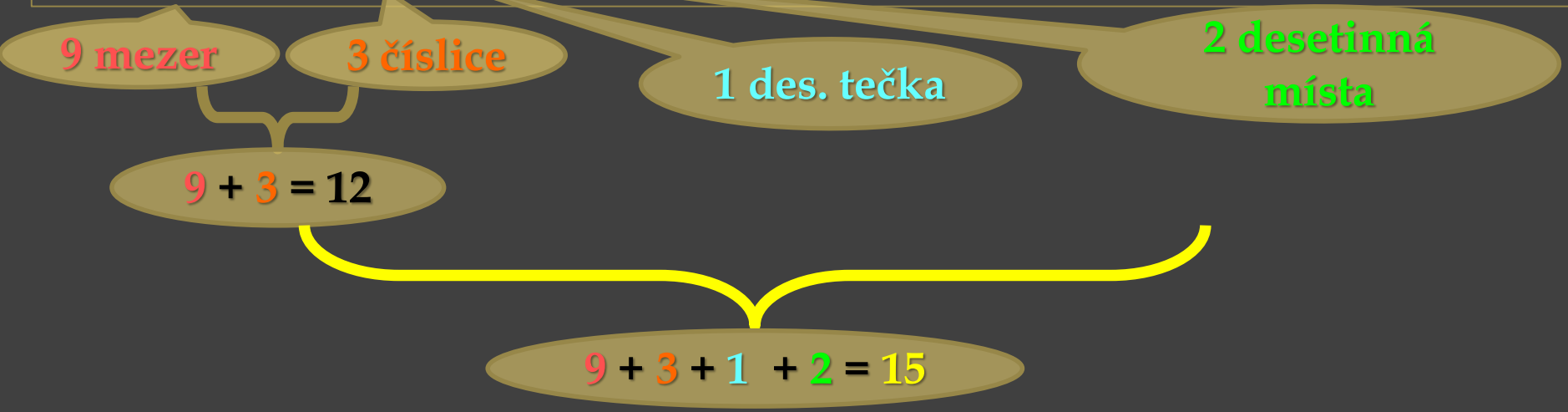
**`fprintf`**

Příklad:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (tj. na celou část zbývá minimálně **12**, **celá část** má **3** číslice =>doplněno **9 mezerami**)

```
fprintf('%15.2f\n', x);
         123.35
```



# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis desetinného čísla na celkem 15 znaků včetně desetinné tečky a z toho budou 2 desetinná místa (doplněno 9 mezerami)

```
fprintf('%15.2f\n', x);
          123.35
```

– a teď doplněno 9 nulami

```
fprintf('%015.2f\n',x);
000000000123.35
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis desetinného čísla na celkem 15 znaků včetně desetinné tečky a z toho budou 2 desetinná místa (doplněno 9 mezerami)

```
fprintf('%15.2f\n', x);
_________123.35
```

– a teď doplněno 9 nulami

```
fprintf('%015.2f\n',x);
000000000123.35
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (doplněno **9 mezerami**)

```
fprintf('%15.2f\n', x);
          123.35
```

– a teď doplněno **9 nulami**

```
fprintf('%015.2f\n',x);
000000000123.35
```

```
fprintf('%06.2f\n',x);
123.35
```

– **nula** není doplněna, už je vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (doplněno **9 mezerami**)

```
fprintf('%15.2f\n', x);
          123.35
```

– a teď doplněno **9 nulami**

```
fprintf('%015.2f\n',x);
000000000123.35
```

3 + 1 + 2 = 6

```
fprintf('%06.2f\n',x);
123.35
```

– **nula** není doplněna, už je vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (doplněno **9 mezerami**)

```
fprintf('%15.2f\n', x);
          123.35
```

– a teď doplněno **9 nulami**

```
fprintf('%015.2f\n',x);
000000000123.35
```

3 + 1 + 2 = 6

```
fprintf('%06.2f\n',x);
123.35
```

– **nula** není doplněna, už je vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

```
fprintf('%04.2f\n',x);
123.35
```

– vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (doplněno 9 **mezerami**)

```
fprintf('%15.2f\n', x);
          123.35
```

– a teď doplněno **9 nulami**

```
fprintf('%015.2f\n',x);
000000000123.35
```

3 + 1 + 2 = 6

```
fprintf('%06.2f\n',x);
123.35
```

– **nula** není doplněna, už je vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

```
fprintf('%04.2f\n',x);
123.35
```

4 - 1 - 2 = 1

– vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

# Funkce pro vstup a výstup

## Formátovaný textový výstup

## fprintf

Pokračování příkladu:

```
x = 123.3456;
```

– výpis **desetinného** čísla na celkem **15** znaků včetně **desetinné tečky** a z toho budou **2 desetinná místa** (doplněno 9 **mezerami**)

```
fprintf('%15.2f\n', x);
          123.35
```

– a teď doplněno **9 nulami**

```
fprintf('%015.2f\n',x);
000000000123.35
```

3 + 1 + 2 = 6

```
fprintf('%06.2f\n',x);
123.35
```

– **nula** není doplněna, už je vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

```
fprintf('%04.2f\n',x);
123.35
```

4 - 1 - 2 = 1

– vypsáno **6** znaků (**2** desetinná místa, **1** tečka a **3** znaky celku)

**– uvedené číslo říká "alespoň 4 znaky celkem", nesmí být oříznuta delší celá část.**

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

Příklad:

**+** značí **vždy** tisknout **znaménko** **+** nebo **–**, např.: `%+d`, `%+f`

```
c = 7; d = -34; x = 123.3456;
fprintf('%+d %+d %+.1f\n', c, d, x);
+7 -34 +123.3
```

**-** znamená **zarovnat doleva**, např.: `%-8.3f`

```
fprintf('%-8.3f\n', x);
123.346
fprintf('%-15.3f\n', x);
123.346
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

Příklad:

**+** značí **vždy** tisknout **znaménko** **+** nebo **–**, např.: **%+d**, **%+f**

```
c = 7; d = -34; x = 123.3456;
fprintf('%+d %+d %+.1f\n', c, d, x);
+7 -34 +123.3
```

**-** znamená **zarovnat doleva**, např.: **%-8.3f**

```
fprintf('%-8.3f\n', x);
123.346␣
fprintf('%-15.3f\n', x);
123.346␣␣␣␣␣␣␣␣
```

před nejsou mezery

*Pozn.* bez znaménka – doplnění **mezerami** vlevo (před číslem)

```
fprintf('%15.3f\n', x);
␣␣␣␣␣␣␣␣123.346
fprintf('%8.3f\n', x);
␣123.346
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

Příklad:

```
a = 234; b = 398; c = 25;
fprintf('V desitkove soustave: %d %d %d\n', a, b, c);
V desitkove soustave: 234 398 25

fprintf('V osmickove soustave: %o %o %o\n', a, b, c);
V osmickove soustave: 352 616 31

fprintf('V 16kove soustave: %x %x %x\n', a, b, c);
V 16kove soustave: ea 18e 19

fprintf('V 16kove soustave: %X %X %X\n', a, b, c);
V 16kove soustave: EA 18E 19


fprintf('OCT %o, DEC %d\n', 25, 25);
OCT 31, DEC 25
```

# Funkce pro vstup a výstup

## Formátovaný textový výstup

**fprintf**

Příklad:

dekadická, decimální

```
a = 234; b = 398; c = 25;
fprintf('V desitkove soustave: %d %d %d\n', a, b, c);
V desitkove soustave: 234 398 25
```

oktalová

```
fprintf('V osmickove soustave: %o %o %o\n', a, b, c);
V osmickove soustave: 352 616 31

fprintf('V 16kove soustave: %x %x %x\n', a, b, c);
V 16kove soustave: ea 18e 19

fprintf('V 16kove soustave: %X %X %X\n', a, b, c);
V 16kove soustave: EA 18E 19
```

hexadecimální

```
fprintf('OCT %o, DEC %d\n', 25, 25);
OCT 31, DEC 25
```