# Seminář 6

## Oběktově orientované programování

- Cell array
- Struktury
- Objekty

## Cell arrays

`Cell array` je datový typ, kde můžeme uchovávat pod jedním názvem více hodnot, obdobně jako v matici. Hlavní rozdíl je v tom, že jednotlivé elementy mohou být různého datového typu. Jednotlivým elementům říkáme `cell`.

Vytváří se výčtem hodnot uzavřených ve složených závorkách.

```
c = {42 randi(5,[2,2]) "ahoj"}
```

```
c = 1×3 cell
```

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 42 | [4,5;1,5] | "ahoj" |

K jednotlivým datům přistupujeme pomocí indexů (obdobně jako u matic), které píšme do složených závorek.

```
c{3}
```

```
ans =
"ahoj"
```

Stejně jako matice jednotlivé prvky v řádku oddělujeme mezerou nebo čárkou, řádky oddělujeme středníkem. Indexujeme pomocí dvou indexů

```
c2 = {1 2 3; 'ahoj' ' ' 'svete'}
```

```
c2 = 2×3 cell
```

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 1 | 2 | 3 |
| 2 | 'ahoj' | ' ' | 'svete' |

```
c2{2,1}
```

```
ans =
'ahoj'
```

Klasické indexování pomocí kulatých závorek vrátí `cell`.

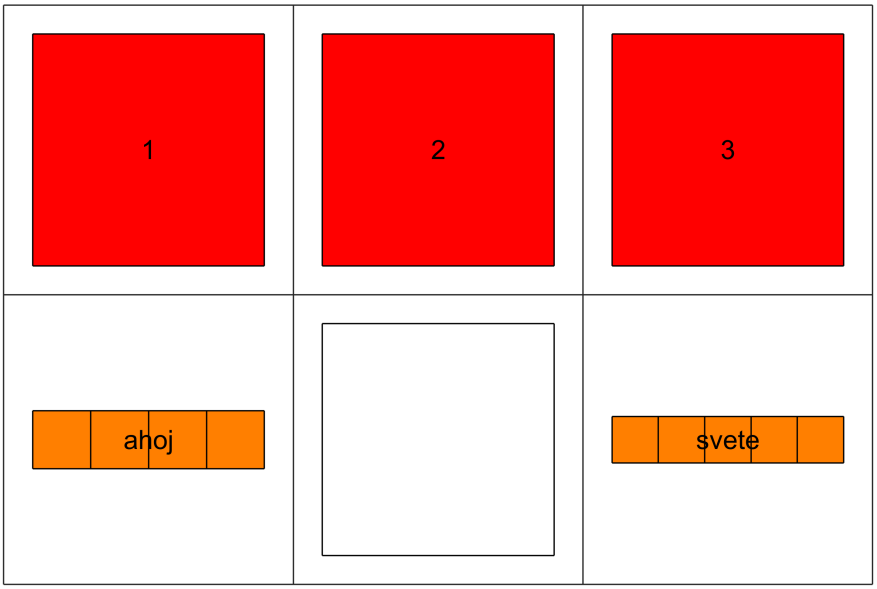
```
c2(2,1)
```

```
ans = 1×1 cell array
    {'ahoj'}
```

Cell array je ve skutečnosti pole ukazatelů (cell je ukazatel)

Grafické znázornění `cell array` pomocí funkce `cellplot()`.

```
cellplot(c2)
```




## Převod mezi datovými typy a `cell array`

`cell2mat()`, `cell2struct()`, `cell2table()`, `mat2cell()`, `num2cell()`, `struct2cell()`, `table2cell()`.

## Prealokace paměti

Pomocí funkce `cell()` nebo vytvořením prázného `cell array`. Následující dva přístupy jsou ekvivalentní.

```
c3 = cell(2,5);
c4{2,5} = [];
```

Nejčastěji se cell array používají pro uchovávání řetězců (ne stringů).

```
% vysledkem je jeden retezec (jedno pole)
M = ['ahoj' ' ' 'svete']
```

```
M =
'ahoj svete'
```

```
% takto nefunguje
N(1) = 'ahoj';
```

```
Unable to perform assignment because the indices on the left side are not compatible with the size of the right side.
```

```
N(2) = ' ';
N(3) = 'svete';
N
```

Nefunguje ani

```
M = ['ahoj'; ' '; 'svete']
```

**Proč?**

```
% 3 retezce
M1 = {'ahoj' ' ' 'svete'}
```

```
M1 = 1×3 cell
'ahoj'      ' '         'svete'
```

```
N1{1} = 'ahoj';
N1{2} = ' ';
N1{3} = 'svete';
N1
```

```
N1 = 1×3 cell
'ahoj'      ' '         'svete'
```

*Vytvořte `cell array`, které jako první položku bude obsahovat název grafu, jako druhou náhodná data (vektor obsahující 20 hodnot), Vykreslete tento graf spolu s názvem grafu (title).*

## Struktury

Struktury si můžeme představit jako pole s pojmenovanými položkami. Jedlotlivé položky mohou být různého datového typu. Pro přístup k jednotlivým položkám používáme operátor `.`

Pomocí něj také strukturu vytváříme. Viz následující příklad pro strukturu bod.

```
bodA.x = 10;
bodA.y = 0;

bodA
```

```
bodA = struct with fields:
```

```
    x: 10
    y: 0
```

```
whos bod

bodA.x
```

```
ans = 10
```

*Poklepejte 2x na proměnnou bod ve workspace.*

Jednotlivé položky struktury mohou být také struktury. Například úsečka.

```
bodB.x = 10;
bodB.y = 5;

usecka.A = bodA;
usecka.B = bodB;

usecka
```

```
usecka = struct with fields:
    A: [1×1 struct]
    B: [1×1 struct]
```

```
usecka.A
```

```
ans = struct with fields:
    x: 10
    y: 0
```

```
usecka.A.x
```

```
ans = 10
```

Všechny hodnoty můžeme nastavit najednou pomocí příkazu `struct()`. Jako argumenty se střídají názvy a hodnoty.

```
bodC = struct('x', 50, 'y', 0)
```

```
bodC = struct with fields:
    x: 50
    y: 0
```

Můžeme vytvářet vektory/matice, které obsahuj struktury,

```
matice = [bodA bodB bodC]
```

```
matice = 1×3 struct
```

| Fields | x | y |
|---|---|---|
| 1 | 10 | 0 |
| 2 | 10 | 5 |
| 3 | 50 | 0 |

```
matice(1)
```

```
ans = struct with fields:
    x: 10
    y: 0
```

```
matice(1).x
```

```
ans = 10
```

*Vytvořte strukturu, ve které se bude uchovávat název grafu a data (náhodná data o velikosti 20 hodnot). Vykreslete tento graf spolu s názvem grafu.*

Nevýhoda struktur může být to, že nové položky vytváříme pomocí tečkového operátoru. Pokud bychom měli překlep v názvu položky, tak matlab nezahlásí chybu, vytvoří novou položku.

Předpokládejme, že chceme bodu `bodC` nastavit souřadnici y na 10 a máme přeplou klávesnici na českou.

```
bodC.z = 10
```

```
bodC = struct with fields:
    x: 50
    y: 0
    z: 10
```

```
bodC
```

```
bodC = struct with fields:
    x: 50
    y: 0
    z: 10
```

# Objekty

Řešením může být nepužívat struktury, ale objekty.

## Vytvoření třídy

class definition file -- .m název shodný s názvem třídy.

**syntaxe**:

```
classdef nazev

% popis třídy
```

```
% detailní popis

...

end
```

Uvnitř definujeme vlastnosti (`properties`) a metody (`methods`)

*Podívejme se na třídu `bod`, kde definujeme jen dvě vlastnosti `x` a `y`.*

## Vytvoření objektu

```
bodD = bod;
whos bodD
```

```
  Name      Size            Bytes  Class    Attributes

  bodD      1x1                 8  bod
```

```
% vypis objektu
bodD
```

```
bodD =
  bod with properties:

    x: []
    y: 0
```

K vlastnostem přistupujeme pomocí operátoru `.`

```
bodD.x = 10;
bodD
```

```
bodD =
  bod with properties:

    x: 10
    y: 0
```

*Zkuste nastavit vlastnost, která v objektu neexistuje (bodD.z). Viz odstrašující příklad u struktur.*

## Oprávnění

Jednotlivým vlastnostem můžeme nastavit, zda je možné je modifikovat, zda jsou skryté nebo se dynamicky mění jejich hodnota. Za `properties` napíšeme typ vlastnosti.

**syntaxe**:

```
properties (Atribut1 = hodnota1, Atribut2 = hodnota2,...)

...

end
```

**Více o atributech vlastností:**

https://uk.mathworks.com/help/matlab/matlab_oop/property-attributes.html

*Podívejte se na definici třídy `trida`.*

```
t = trida;
whos t
```

```
  Name      Size            Bytes  Class    Attributes

  t         1x1                 8  trida
```

```
t
```

```
t =
  trida with properties:

    konstanta: 20
      verejna: []
```

```
t.konstanta = 5
```

```
Unable to set the 'konstanta' property of class 'trida' because it is read-only.
```

```
t.verejna = 5
```

```
t =
  trida with properties:

    konstanta: 20
      verejna: 5
```

```
t.privatni
```

```
No public property 'privatni' for class 'trida'.
```

## Metody/operace

**syntaxe**:

```
methods (Atribut1 = hodnota1, Atribut2 = hodnota2,...)

function vystupni_argumenty = method1(obj,vstupni_argumenty)

...

end

end
```

**Konstruktor**

```
function obj = nazev_tridy(vstupni_argumenty)
```

```
obj.vlastnost1 = hodnota1;

...

end
```

*Ve třídě `trida` vytvořte konstruktor a metodu, která vypíše hodnotu vlastnosti `privatni`. (Řešení je např. v definici `trida2`).*

**Vytvoření objektu a volání funkcí**

```
t2 = trida2(10,10)
```

```
t2 =
  trida2 with properties:

    konstanta: 20
      verejna: 10
```

```
% volani funkce vypis_privatni
t2.vypis_privatni
```

```
ans = 10
```

```
vypis_privatni(t2)
```

```
ans = 10
```

```
% volani funkce prijimajici argumenty obj a h
t2.pricti(2)
```

```
ans = 12
```

```
pricti(t2,2)
```

```
ans = 12
```

*Při vytváření objektu musíme respektovat počet argumentů, které konstruktor očekává. V minulé hodině jsme se bavili o variadických funkcích. Upravte konstruktor tak, že pokud nebude přijímat žádnou argument, nenastaví nic, pokud jeden nastaví se hodnota vlastnosti `verejna` (`privatni` ma defaultni hodnotu), pokud dvě pak se nastaví jak `verejna`, tak `privatni`.*

Obecně by v konstruktoru měla být i validace dat.

Stejně jako u vlastností i u metod můžeme nastavit různé atributy. **Více**:

https://uk.mathworks.com/help/matlab/matlab_oop/method-attributes.html

Můžeme nastavit i atributy třídy. **Více**:

https://uk.mathworks.com/help/matlab/matlab_oop/class-attributes.html

Kromě vlastností a metodu můžeme vytvářet i události (`events`). **Více**:

https://uk.mathworks.com/help/matlab/matlab_oop/event-attributes.html

Funkce nemusíme definovat v `class definition file`. Je možné je uložit do složky `@nazev_tridy` a mohou to být samostatné soubory. Ve složce `private` jsou privátní metody. `class definition file` musí být také v této složce.

Podívejte se na třídu bod2.

```
bodE = bod2(1,2,0)
```

```
bodE =
  bod2 with properties:

    x: 1
    y: 2
    z: 0
```

```
bodE.delka
```

```
ans = 1
```

```
%bodE.privatni
bodE.volani_privatni
```

```
ans = 3
```

## Dědičnost

**syntaxe**:

```
classdef nazev < nazev_rodice

...

end
```

Pro vícenásobnou dědičnost

```
classdef nazev < nazev_rodice1 & nazev_rodice2 & nazev_rodice3

...

end
```