Sets in Matlab

A small detour to learn some usefull things in Matlab

Sets of words and Matlab

- We need to know how to have sets of words in Matlab
  - Sets of different sizes
  - Words in each set with different sizes

Storing More Than Numbers

- MATLAB matrices store numeric results

- **What about words, names, strings?**
- **What about arrays of arrays?**
- **What about Sets ?**

- MATLAB provides more containers to store data
  - **Character arrays**
  - **Cell arrays**
  - **Structures**

Character Arrays

- Examples:


```
» C = 'Hello';          %C is a 1x5 character array.
» D = 'Hello there';    %D is a 1x11 character array.
» A = 43;               %A is a 1x1 double array.
» T = 'How about this character string?'
```

```
» size(T)
ans =

     1    32
```

How are Characters Stored?

- Character arrays are similar to vectors, except:
  - Each cell contains a single digit
- Example

```
» u = double(T)   % double is a dedicated function.
» char(u)         % performs the opposite function.
```

- Exercise

```
» a = double('a') → código ascii de 'a'
» char(a) → numero para caracter
```

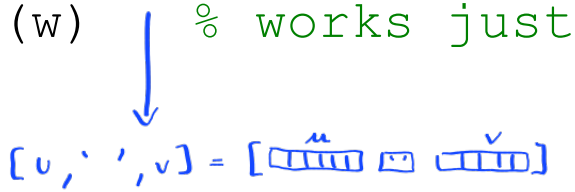
- Questions: What is the numerical value of ‘a’ and what does it mean?

Manipulating Strings

- Strings can be manipulated like arrays.

- Examples

```
» u = T(16:24)
» u = T(24:-1:16)
» u = T(16:24)'
» v = 'I can''t find the manual!'  %  Note quote in
  string
» u ='If a woodchuck could chuck wood,';
» v = 'how much wood could a woodchuck chuck?';
» w = [u,' ', v] % string concatenation in Matlab
» disp(w)        % works just like for arrays
```



Cell Arrays

→ Guarda informação diversa!

- Cell arrays are containers for “collections” of data of any type stored in a common container.

- Cell arrays are like a wall of PO boxes, with each PO box containing its own type of information.
- When mail is sent to a PO box the PO box number is given. Similarly, each cell in a cell array is indexed.
- Cell arrays are created using cell indexing in the same way that data in a table or an array is created and referenced

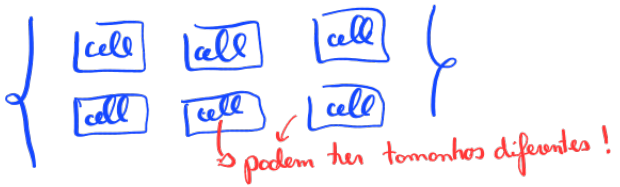
- **The difference is the use of curly braces { }.**

Matrix of matrices

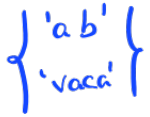
- Cell arrays are matrix of matrices

- Example:

```
x=[1:5]; y = floor(2.*randn(1,5));
z = [100:-20:20];
M = [x; y; z]

 c = {M  M+M; M(:,1) M(3,:) }
```




```
c =
   2×2 cell array

   {3×5 double}    {3×5 double}
   {3×1 double}    {1×5 double}
```



Cell array example

- create in the same way as arrays but use (curly) braces

```
>> a = { i 5:-1:2 'carrots'; magic(2) 77 NaN }
```

troca de linha

```
a =
    [0 + 1.0000i]    [1x4 double]    'carrots'
     [2x2 double]    [       77]    [    NaN]
```

Create empty cell array

Using `cell()` function:

```
a = cell( rows, columns)
```

```
a = cell( 3, 6 )
a =
    []    []    []    []    []    []
    []    []    []    []    []    []
    []    []    []    []    []    []

whos a
  Name        Size                Bytes  Class
  a           3x6                    72  cell
```

Cell Array Access

- Cell arrays look a lot like arrays but they cannot generally be manipulated the same way.

- Cell arrays should be considered more as data "containers" and must be manipulated accordingly.
  - *Cell arrays cannot be used in arithmetic computations like arrays can, e.g., + - * / ^*

Addressing Cell Arrays

- `A(i,j) = {x}` → célula // Sem operações!

  this is called CELL INDEXING

- `A{i,j} = x` → conteúdo //

  this is called CONTENT ADDRESSING

- either can be used, but be careful…

Examples

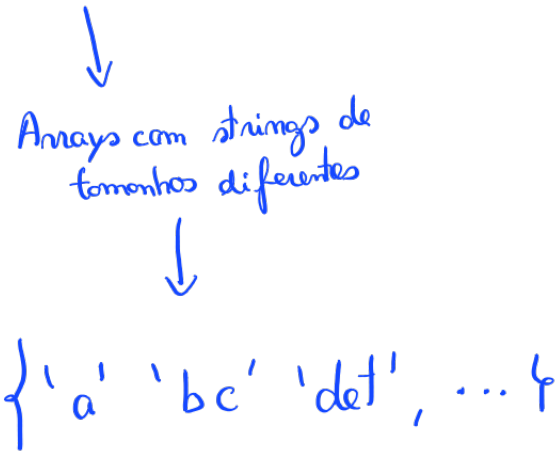
```
first = 'Hello';
second = {'hello','world','from','me'};

third(1,1) = {'happy'};  % Cell indexing
third{2,1} = 'birthday';  % Content addressing
third{3,1} = 40;
```

- What will we obtain from ?

```
>> third → {1×'happy', 'birthday', 40}
>> third(1,1), third{1,1}
>> third(2,1), third{2,1}
>> third(3,1), third{3,1}
```

Cell Arrays of Strings

- All rows in a string array MUST have the same number of columns … this is a problem for representing our sets of words
  - An many other problems



- Solution?
- **Cell arrays**

Exercise

```
C = {'How';'about';'this for a';'cell array of strings?'}


size(C)

C(2:3)

C([4,3,2,1])

[a,b,c,d] = deal(C{:})
```

Examples

```
» C = cell(2,3) % Defines C to be a cell array
» C(1,1) = {'This does work'} % ( ) refer to PO Box

» C{2,3} = 'This works too'    % { } refers to
  contents

Try:

» A = cell(1,3) % Note 1 x 3

» A = {'My' , 'name', 'is' , 'Burdell'} %  Note 1 x 4

» A = {'My'; 'name'; 'is' ; 'Burdell'}

Get more info:
» help lists
```

Set Operations

- Matlab provides several functions for set operations !

intersect	Set intersection of two arrays
ismember	Array elements that are members of set array
setdiff	Set difference of two arrays
setxor	Set exclusive OR of two arrays
union	Set union of two arrays
unique	Unique values in array
ismembertol	Members of set within tolerance
uniquetol	Unique values within tolerance
join	Combine two tables or timetables by rows using key variables
innerjoin	Inner join between two tables or timetables
outerjoin	Outer join between two tables or timetables

join

Example

A={'a' 'e' 'i' 'o' 'u'} → cell array

B={'a','b','c','d','e'}

C=intersect(A,B) % o que dará ?

D=union(A,B)

ismember(A(1),C) → membro ou não !

ismember(A,D) % o que dará ?

```
ans =
     1     1     1     1     1
```

Some Useful functions

```
» iscellstr(A) % logical test for a cell
  array of strings
» ischar(A)  % logical test for a string
  array

» celldisp(B) % recursively displays cell
  array, i.e., if content a cell array,
  also displays its content

» cellstr(B)
```

Use help to get information on each of these functions …

Some Useful functions

```
» cellplot(B)   % displays in figure window drawing of 1D or 2D cell array

» cell2mat(B)   %  convert a cell array of numbers to a numerical array

» num2cell( A ) % convert an array of numbers to a cell array

» cellfun( A )  %  applies a specified function to the content of every element of a cell array
```

Structures

- Numeric, character and cell arrays all reference the individual elements by number
- Structures reference individual elements within each row (called "*fields*") by name.
- To access these fields, the dot "." notation is used.
- Assignment is as follows:

```
structurename.fieldname =
  datatype;
```

Creating a Structure...

- Let's create a simple structure:

```
person.firstname = 'António';
person.lastname = 'Teixeira';
person.address1 = 'DETI/IEETA,
University of Aveiro';
person.city = 'Aveiro';
person.zip = '3810-193 AVEIRO';
```

...

```
person =

    firstname: ‘António'
     lastname: ‘Teixeira'
     address1: [1x32 char]
         city: 'Aveiro'
          zip: '3810-193 AVEIRO'
```

More on Structures...

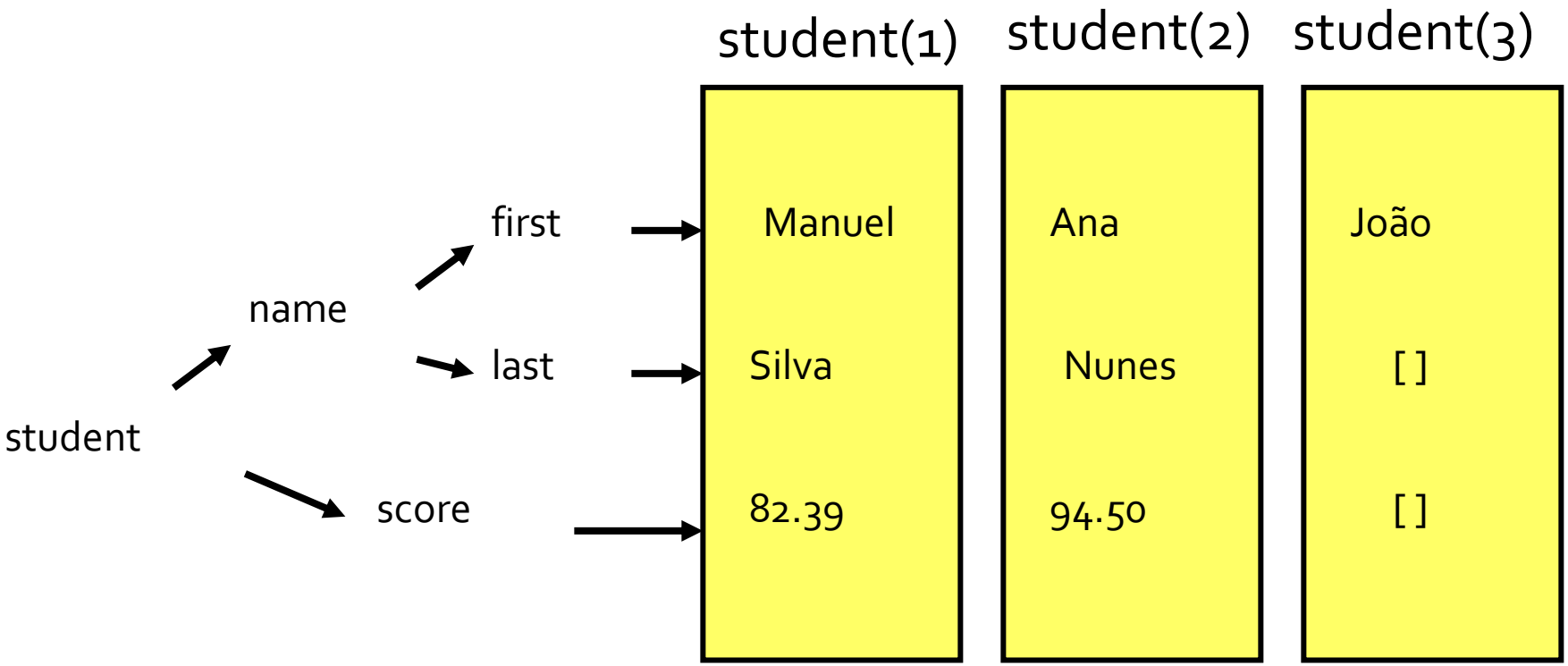
- A structure can have a field that is a structure itself.
- A structure array is that which contains more than one record for each field name.
- As the structure array is expanded (more records are created), all unassigned fields are filled with an empty matrix.
- All structures have the same number of fields and elements in each field.

Example

```
student(1).name.first = 'Manuel';
student(1).name.last = 'Silva';
Student(1).score = 82.39;

student(2).name.first = 'Ana';
student(2).name.last = 'Nunes';
student(2).score = 94.50;

student(3).name.first = 'João';
…
```

Example (cont.)

! Muito útil !



Sources used

- PPT on “Strings, Cell Arrays and Structures” of AE6382-9 Design Computing course, Georgia Tech, 2006
- PPT “Matlab Cell Arrays” by Greg Reese, Miami University, 2011
- Chapters 7 and 8 of Duane Hanselman and Bruce Littlefield (2003), “Matlab 6 Curso Completo”, Prentice Hall