Table of Contents

Семинар 4. Контейнеры для работы с разнотипными данными (продолжение)

- Таблицы
- Множества
- Словари
- Использование контейнеров java

Небольшой пример по использованию структур и функции **{:}** из предыдущего семинара

Конструктор объекта типа **struct** имеет следующую форму:

struct("field1_name",value1,...."fieldN_name",valueN)

Для конструкирования структуры с заданными именами полей можно использовать возможности "splat" - функции ячеек.

Для примера создадим стурктуру с именами полей от 'a' до 'y' и значениями в этих полях от 1 до 25

```
clearvars
field_names = arrayfun(@string,'a':'y');
n = numel(field_names);
Name_Values_cell = cell(1,2*n); % создаем пустой массив ячеек,
% в него будут поочередно добавлены имена полей и их значения

% заплоняем имена полей
Name_Values_cell(1:2:end) = cellstr(field_names); % cellstr - преобразует массив string в ячейку массивов char
Name_Values_cell
```

```
Name_Values_cell = 1×50 cell
```

...

	1	2	3	4	5	6	7	8
1	'a'	[]	'b'	[]	'c'	[]	'd'	[]

```
% заполняем содержимое ячеек
Name_Values_cell(2:2:end) = num2cell(1:n); %
Name_Values_cell
```

```
Name_Values_cell = 1×50 cell
```

...

	1	2	3	4	5	6	7	8
1	'a'	1	'b'	2	'c'	3	'd'	4

```
st = struct(Name_Values_cell{:}) % конструктор структур поддерживает произвольное
число аргументов
```

```
st = struct with fields:
    a: 1
    b: 2
    c: 3
    d: 4
    e: 5
    f: 6
    g: 7
    h: 8
    i: 9
    j: 10
    k: 11
    l: 12
    m: 13
    n: 14
    o: 15
    p: 16
    q: 17
    r: 18
    s: 19
    t: 20
    u: 21
    v: 22
    w: 23
    x: 24
    y: 25
```

```
st.y
```

```
ans = 25
```

тип table

Для хранения данных в табличках с именами столбцов

```
clearvars
folder = get_folder()
```

```
folder = 
'E:\projects\matlab-seminar\basics\sem1_4'
```

```
full_file = fullfile(folder,"tbl.xls")
```

```
full_file = 
"E:\projects\matlab-seminar\basics\sem1_4\tbl.xls"
```

```
cell_1 = cell(21,6);
cell_1(2:end,:) = num2cell(rand([20 6]));
cell_1(1,:) = {"а" "ё" "а" "а" "и" "л" };
writecell(cell_1,full_file);
tbl1 = readtable(full_file);
```

```
Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating
variable names for the table. The original column headers are saved in the VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.
```

```
% можно читать таблички из эксель или текстовых файлов с разделителем,
например, .csv
% readtable - высокоуровневая читалка с очень большим набором функций
help("readtable")
```

```
 readtable Create a table by reading from a file.
 
    Use the readtable function to create a table by reading column-oriented
    data from a file. readtable automatically determines the file format
    from its extension as described below.
 
    T = readtable(FILENAME) creates a table by reading from a file, where
    FILENAME can be one of these:
 
        - For local files, FILENAME can be a full path that contains
          a filename and file extension. FILENAME can also be a relative
          path to the current folder, or to a folder on the MATLAB path.
          For example, to import a file on the MATLAB path:
 
             T = readtable("patients.xls");
 
        - For files from an Internet URL or stored at a remote location,
          FILENAME must be a full path using a Uniform Resource Locator
          (URL). For example, to import a remote file from Amazon S3,
          specify the full URL for the file:
 
             T = readtable("s3://bucketname/path_to_file/my_table.xls");
 
          To read tabular data from a web page, specify the URL; if the
          URL points to an HTML resource, but does not end in ".html"
          or ".htm", also specify the file type:
 
             url = "https://www.mathworks.com/matlabcentral/cody/groups/78";
             T = readtable(url,"FileType","html");
 
          For more information on accessing remote data, see "Work with
          Remote Data" in the documentation.
 
    T = readtable(FILENAME,"FileType",FILETYPE) specifies the file type, where
    FILETYPE is one of "text", "delimitedtext", "fixedwidth", "spreadsheet",
    "xml", "html", or "worddocument".
 
    T = readtable(FILENAME,OPTS) creates a table by reading from a file stored
    at FILENAME using the supplied ImportOptions OPTS. OPTS specifies variable
    names, selected variable names, variable types, and other information regarding
    the location of the data.
 
    For example, import a subset of the data in a file:
 
        opts = detectImportOptions("patients.xls");
        opts.SelectedVariableNames = ["Systolic","Diastolic"];
    T = readtable("patients.xls",opts)
```

readtable reads data from different file types as follows:

Text files (delimited and fixed-width):

```
  The following extensions are supported: .txt, .dat, .csv, .log,
                                          .text, .dlm
```

Reading from a delimited text file creates one variable in T for each column in the file. Variable names can be taken from the first row of the file. By default, the variables created are either double, if the column is primarily numeric, or datetime, duration, or text etc. If data in a column cannot be converted to numeric, datetime or duration, the column is imported as text.

Spreadsheet files:

```
  The following extensions are supported: .xls, .xlsx, .xlsb, .xlsm,
                                          .xltm, .xltx, .ods
```

Reading from a spreadsheet file creates one variable in T for each column in the file. By default, the variables created are either double, datetime or text--depending on the type in the file.

readtable converts both empty fields or cells and values which cannot be converted to the expected type to:

- NaN (for a numeric or duration variable),
- NaT (for a datetime variable),
- Empty character vector ('') or missing string (for text variables).

Word documents:

The following extensions are supported: .docx

Reading from a Word document file imports data from a table. Each column in the table creates one variable in T. Variable names can be taken from the first row of the table. By default, the variables created are either double, if the column is primarily numeric, or datetime, duration, or text etc. If data in a column cannot be converted to numeric, datetime or duration, the column is imported as text. The default data type for text import is string.

HTML files:

The following extensions are supported: .html, .xhtml, .htm

Reading from an HTML file imports data from a <TABLE> element. Each column in the table creates one variable in T. Variable names can be taken from the first row of the table. By default, the variables created are either double, if the column is primarily numeric, or datetime, duration, or text etc. If data in a column cannot be converted to numeric, datetime or duration, the column is imported as text. The default data type for text import is string.

XML files:

The following extensions are supported: .xml

Tabular structure present within an XML file:

```
      <table> ----------------------------- Table Node
          <row> -------------------------- Row Node
              <date>2019-07-11</date> ----- Variable Node
              <index>8191</index>
              <name>Lorem</name>
          </row>
          <row>
              <date>2020-01-04</date>
              <index>131071</index>
              <name>Ipsum</name>
          </row>
      </table>

  Reading from an XML file creates one row in T for each repeated node
  in the file that is detected under the table node. Variable names are
  taken from the names of the child nodes under the row nodes in the file.

Name-Value Pairs for ALL file types:
------------------------------------

"FileType"               - Specify the file as "text", "delimitedtext",
                           "fixedwidth", "spreadsheet", "xml", "html",
                           or "worddocument".

"VariableNamingRule"     - A character vector or a string scalar that
                           specifies how the output variables are named.
                           It can have either of the following values:

                           "modify"   Modify variable names to make them
                                      valid MATLAB Identifiers.
                                      (default)
                           "preserve" Preserve original variable names
                                      allowing names with spaces and
                                      non-ASCII characters.

"MissingRule"            - Rules for interpreting missing or
                           unavailable data:
                           "fill"      Replace missing data with the
                                       contents of the "FillValue"
                                       property.
                           "error"     Stop importing and display an
                                       error message showing the missing
                                       record and field.
                           "omitrow"   Omit rows that contain missing
                                       data.
                           "omitvar"   Omit variables that contain
                                       missing data.

"ImportErrorRule"        - Rules for interpreting nonconvertible
                           or bad data:
                           "fill"      Replace the data where errors
                                       occur with the contents of the
                                       "FillValue" property.
                           "error"     Stop importing and display an
                                       error message showing the
                                       error-causing record and field.
                           "omitrow"   Omit rows where errors occur.
                           "omitvar"   Omit variables where errors
                                       occur.

"ReadRowNames"           - Whether or not to import the first variable
                           as row names. Defaults to false.

"TreatAsMissing"         - Text which is used in a file to represent
                           missing data, e.g. "NA".

"TextType"               - The type to use for text variables, specified
```

```
                          as "char" or "string".

"DatetimeType"          - The type to use for date variables, specified
                          as "datetime", "text", or "exceldatenum".
                          Defaults to "datetime".

"WebOptions"            - HTTP(s) request options, specified as a
                          weboptions object.

Name-Value Pairs for TEXT and SPREADSHEET only:
-----------------------------------------------

"Range"                 - The range to consider when detecting data.
                          Specified using any of the following syntaxes:
                          - Starting cell: A string or character vector
                            containing a column letter and a row number,
                            or a 2 element numeric vector indicating
                            the starting row and column.
                          - Rectangular range: A start and end cell separated
                            by colon, e.g. "C2:N15", or a four element
                            numeric vector containing start row, start
                            column, end row, end column, e.g. [2 3 15 13].
                          - Row range: A string or character vector
                            containing a starting row number and ending
                            row number, separated by a colon.
                          - Column range: A string or character vector
                            containing a starting column letter and
                            ending column letter, separated by a colon.
                          - Starting row number: A numeric scalar
                            indicating the first row where data is found.

"NumHeaderLines"        - The number of header lines in the file.

"ExpectedNumVariables"  - The expected number of variables.

"ReadVariableNames"     - Whether or not to expect variable names in
                          the file. Defaults to true.

Name-Value Pairs for TEXT, XML, HTML, and Word documents only:
--------------------------------------------------------------

"DateLocale"         - The locale used to interpret month and day
                       names in datetime text. Must be a character
                       vector or scalar string in the form xx_YY.
                       See the documentation for DATETIME for more
                       information.

"DecimalSeparator"   - Character used to separate the integer part
                       of a number from the decimal part of the
                       number.

"ThousandsSeparator" - Character used to separate the thousands
                       place digits.

Name-Value Pairs for TEXT, XML, and HTML only:
----------------------------------------------

"Encoding"           - The character encoding scheme associated with
                       the file.

Name-Value Pairs for TEXT and XML only:
---------------------------------------

"DurationType"       - The type to use for duration, specified as
```

```
                       "duration" or "text". Defaults to "duration".

"Whitespace"         - Characters to treat as whitespace.

"TrimNonNumeric"     - Whether or not to remove nonnumeric characters
                       from a numeric variable. Defaults to false.

"HexType"            - Set the output type of a hexadecimal
                       variable.

"BinaryType"         - Set the output type of a binary variable.

"CollectOutput"      - Whether or not to concatenate consecutive output
                       of the same MATLAB class into a single array.
                       Defaults to false.

Name-Value Pairs for TEXT, HTML, and Word documents only:
---------------------------------------------------------

"RowNamesColumn"     - The column where the row names are
                       located.

Name-Value Pairs for TEXT only:
-------------------------------

"Delimiter"                 - Field delimiter characters in a delimited
                              text file, specified as a character
                              vector, string scalar, cell array of
                              character vectors, or string array.

"CommentStyle"              - Style of comments, specified as a
                              character vector, string scalar, cell
                              array of character vectors, or string
                              array.

"LineEnding"                - End-of-line characters, specified as a
                              character vector, string scalar, cell
                              array of character vectors, or string
                              array.

"ConsecutiveDelimitersRule" - Rule to apply to fields containing
                              multiple consecutive delimiters:
                              "split"     Split consecutive delimiters
                                          into multiple fields.
                              "join"      Join the delimiters into one
                                          single delimiter.
                              "error"     Ignore consecutive delimiters
                                          during detection (treated as
                                          "split"), but the
                                          resulting read will error.

"LeadingDelimitersRule"     - Rule to apply to delimiters at the
                              beginning of a line:
                              "keep"      Keep leading delimiters.
                              "ignore"    Ignore leading delimiters.
                              "error"     Ignore leading delimiters
                                          during detection, but the
                                          resulting read will error.

"TrailingDelimiterRule"     - Rule to apply to delimiters at the
                              end of a line:
                              "keep"      Keep trailing delimiters.
                              "ignore"    Ignore trailing delimiters.
                              "error"     Ignore trailing delimiters
```

```
                                              during detection, but the
                                              resulting read will error.

    "VariableWidths"            - Widths of the variables for a fixed width
                                  file.

    "EmptyLineRule"             - Rule to apply to empty lines in the file:
                                  "skip"      Skip empty lines.
                                  "read"      Read empty lines.
                                  "error"     Ignore empty lines during
                                              detection, but the resulting
                                              read will error.

    "VariableNamesLine"         - The line where the variable names are
                                  located.

    "PartialFieldRule"          - Rule to handle partial fields in the data:
                                  "keep"      Keep the partial field data
                                              and convert the text to the
                                              appropriate data type.
                                  "fill"      Replace missing data with the
                                              contents of the "FillValue"
                                              property.
                                  "omitrow"   Omit rows that contain
                                              partial data.
                                  "omitvar"   Omit variables that contain
                                              partial data.
                                  "wrap"      Begin reading the next line
                                              of characters.
                                  "error"     Ignore partial field data
                                              during detection, but the
                                              resulting read will error.

    "VariableUnitsLine"         - The line where the variable units are
                                  located.

    "VariableDescriptionsLine"  - The line where the variable descriptions
                                  are located.

    "ExtraColumnsRule"          - Rule to apply to extra columns of data
                                  that appear after the expected variables:
                                  "addvars"   Creates new variables to
                                              import extra columns. If there
                                              are N extra columns, then import
                                              new variables as "ExtraVar1",
                                              "ExtraVar2",..., "ExtraVarN".
                                  "ignore"    Ignore the extra columns of
                                              data.
                                  "wrap"      Wrap the extra columns of
                                              data to new records.
                                  "error"     Display an error message and
                                              abort the import operation.

    Name-Value Pairs for SPREADSHEET only:
    --------------------------------------

    "UseExcel"                  - Whether or not to read the spreadsheet
                                  file using Microsoft(R) Excel(R) on
                                  Windows(R):
                                  true  - Opens an instance of Microsoft Excel
                                          to read the file on a Windows system
                                          with Excel installed.
                                  false - Does not open an instance of Microsoft
                                          Excel to read the file. This is the
```

```
                                      default setting.

"Sheet"                     - The sheet from which to read the table.

"DataRange"                 - Where the table data is located.

"RowNamesRange"             - Where the row names are located.

"VariableNamesRange"        - Where the variable names are located.

"VariableUnitsRange"        - Where the variable units are located.

"VariableDescriptionsRange" - Where the variable descriptions are
                              located.

Name-Value Pairs for HTML and Word documents only:
--------------------------------------------------

"TableIndex"                - Integer selection which table to extract.

"TableSelector"             - XPath expression that selects the table
                              to extract.

"VariableNamesRow"          - The row where the variable names are
                              located.

"VariableUnitsRow"          - The row where the variable units are
                              located.

"VariableDescriptionsRow"   - The row where the variable descriptions
                              are located.

"EmptyRowRule"              - Rule to apply to empty lines in the file:
                              "skip"      Skip empty lines.
                              "read"      Read empty lines.
                              "error"     Ignore empty lines during
                                          detection, but the resulting
                                          read will error.

"EmptyColumnRule"           - Rule to apply to empty columns in the file:
                              "skip"      Skip empty columns.
                              "read"      Read empty columns.
                              "error"     Error on empty columns.

Name-Value Pairs for XML only:
------------------------------

"RowNodeName"                  - Node name which delineates rows of
                                 the output table.

"RowSelector"                  - XPath expression that selects the XML
                                 Element nodes which delineate rows of
                                 the output table.

"VariableNodeNames"            - Node names which will be treated as
                                 variables of the output table.

"VariableSelectors"            - XPath expressions that select the XML
                                 Element nodes to be treated as variables
                                 of the output table.

"TableNodeName"                - Name of the node which contains table
                                 data. If multiple nodes have the same
                                 name, readtable uses the first node
                                 with that name.

"TableSelector"               - XPath expression that selects the XML
                                 Element node containing the table data.

"VariableUnitsSelector"       - XPath expression that selects the XML
                                 Element nodes containing the variable
                                 units.

"VariableDescriptionsSelector" - XPath expression that selects the XML
                                 Element nodes containing the variable
                                 descriptions.

"RowNamesSelector"            - XPath expression that selects the XML
                                 Element nodes containing the row names.

"RepeatedNodeRule"            - Rule for managing repeated nodes in a
                                 given row of a table:
                                 "addcol"     Add a column for each
                                              repeated node.
                                 "ignore"     Ignore repeated nodes.
                                 "error"      Ignore repeated nodes
                                              during detection, but the
                                              resulting read will error.

"ImportAttributes"            - Import XML node attributes as variables
                                 of the output table. Defaults to true.

"AttributeSuffix"             - Suffix to append to all output table
                                 variable names corresponding to
                                 attributes in the XML file. Defaults
                                 to "Attribute".

"RegisteredNamespaces"        - The namespace prefixes that are mapped
                                 to namespace URLs for use in selector
                                 expressions.

Name-Value Pairs supported with Text and Spreadsheet Import Options OPTS:
-------------------------------------------------------------------------

    Supported for all file types:
      "WebOptions" -   HTTP(s) request options, specified as a
                       weboptions object.

These have slightly different behavior when used with import options:

    T = readtable(FILENAME, OPTS, "Name1", Value1, "Name2", Value2, ...)

      "ReadVariableNames" true  - Reads the variable names from the
                                  opts.VariableNamesRange or opts.VariableNamesLine
                                  location.
                          false - Uses variable names from the import options.

      "ReadRowNames"      true  - Reads the row names from the opts.RowNamesRange
                                  or opts.RowNamesColumn location.
                          false - Does not import row names.

    Text only parameters:
      "DateLocale" - Override the locale used when importing dates.
      "Encoding"   - Override the encoding defined in import options.

    Spreadsheet only parameters:
      "Sheet"      - Override the sheet value in the import options.
      "UseExcel"   - Same behavior as READCELL without import options.
See also writetable, readtimetable, readmatrix, readcell, table, detectImportOptions

Documentation for readtable
```

```
% функции, которые работают с объектами типа таблица
methods(tbl1)
```

```
Methods for class table:

abs           acos          acosd         acosh         acot          acotd         acoth         acsc          acs
```

```
tbl1.Properties.VariableNames = {'name' 'a1' 'a2' 'a3' 'a4' 'a5'}
```

tbl1 = 20×6 table

	name	a1	a2	a3	a4	a5
1	0.0974	0.3913	0.5551	0.1277	0.5451	0.6188
2	0.3239	0.8838	0.9277	0.8301	0.3967	0.5791
3	0.7422	0.3928	0.9631	0.2053	0.4661	0.6015
4	0.7053	0.3115	0.1312	0.2982	0.8596	0.4623
5	0.1562	0.9740	0.3327	0.7010	0.0083	0.3925
6	0.4454	0.8265	0.5781	0.0976	0.8565	0.0915
7	0.6564	0.2700	0.2750	0.6080	0.6196	0.9858
8	0.0865	0.7921	0.5619	0.6125	0.0488	0.3789
9	0.0355	0.1016	0.8469	0.9810	0.2501	0.9664
10	0.0578	0.6599	0.7820	0.7856	0.2255	0.8622
11	0.9526	0.1942	0.5140	0.3219	0.0351	0.8792
12	0.8647	0.1469	0.3851	0.2038	0.9486	0.5805
13	0.6102	0.6915	0.5478	0.0069	0.6395	0.7336
14	0.1591	0.7253	0.8693	0.9778	0.8950	0.0405

⋮

```
mean_val = mean(tbl1(:,2:end)) % среднее значение
```

mean_val = 1×5 table

	a1	a2	a3	a4	a5
1	0.5322	0.5504	0.4725	0.4923	0.5476

```
standard_deviation=std(tbl1(:,2:end)) % среднее значение
```

standard_deviation = 1×5 table

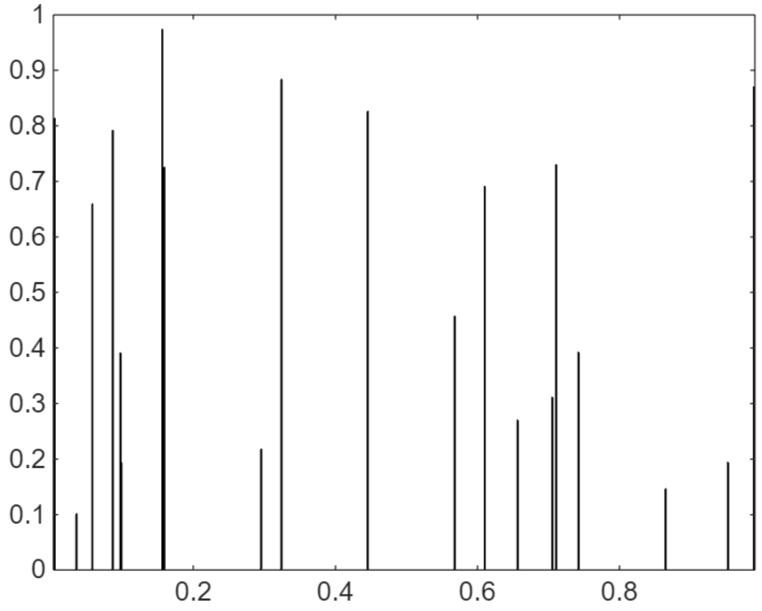
	a1	a2	a3	a4	a5
1	0.2910	0.2672	0.3147	0.3287	0.3030

```
summary(tbl1)
```

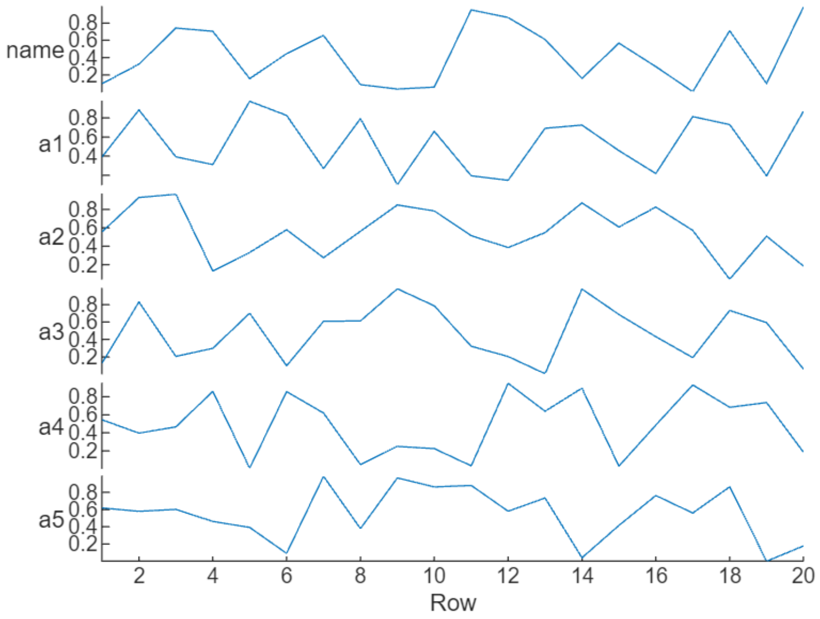
```
Variables:

    name: 20×1 double

        Properties:
            Description:  a
        Values:

            Min       0.0046851
            Median      0.38467
            Max         0.98901

    a1: 20×1 double

        Properties:
            Description:  ё
        Values:

            Min       0.10157
            Median    0.55872
            Max       0.97398

    a2: 20×1 double

        Properties:
            Description:  a
        Values:

            Min       0.042565
            Median     0.55851
            Max        0.96309

    a3: 20×1 double

        Properties:
            Description:  a
        Values:

            Min       0.0068895
            Median      0.51206
            Max         0.98101

    a4: 20×1 double

        Properties:
            Description:  и
        Values:

            Min       0.0082674
            Median      0.51567
            Max         0.94855

    a5: 20×1 double

        Properties:
            Description:  л
        Values:
```

```
        Min       0.0011639
        Median      0.57978
        Max         0.98576
```

```
bar(tbl1.name,tbl1.a1)
```



```
stackedplot(tbl1)
```



```
%tbl2 = table([1:4]',ones(4,3,2),eye(4,2))  - элементы разной размерности
%работают но не отображаюстя в LiveScript
```

тип containers.Map

Для хранения разнородных данных по имени (ключу)

```
clearvars
M = containers.Map('KeyType','char','ValueType','double')
```

```
M = 

  Map with properties:

        Count: 0
      KeyType: char
    ValueType: double
```

```
M("a")=10
```

```
M = 
  Map with properties:

        Count: 1
      KeyType: char
    ValueType: double
```

```
methods(M)
```

```
Methods for class containers.Map:

Map       disp      isKey     isempty   keys      length    remove    size      values    

Static methods:

empty     

Methods of containers.Map inherited from handle.
```

тип dictionary (рекомендуется вместо containers.Map)

Для хранения и получения данных по "ключу"

```
clearvars
d =  dictionary(["sin" "cos" "tan"],{@sin, @cos, @tan})
```

```
d = 

  dictionary (string ⍰ cell) with 3 entries:

    "sin" ⍰ {@sin}
    "cos" ⍰ {@cos}
    "tan" ⍰ {@tan}
```

```
d("sin")
```

```
ans = 1×1 cell array
    {@sin}
```

```
d1 = d("sin")
```

```
d1 = 1×1 cell array
```

```
    {@sin}
```

```
d1{1}(pi)
```

```
ans = 1.2246e-16
```

```
d("cot") = {@cot}
```

```
d =

  dictionary (string ⍰ cell) with 4 entries:

    "sin" ⍰ {@sin}
    "cos" ⍰ {@cos}
    "tan" ⍰ {@tan}
    "cot" ⍰ {@cot}
```

```

d2 = dictionary({[false true true] [true false true] [true true false]},{@sin,
@cos, @tan})
```

```
d2 =

  dictionary (cell ⍰ cell) with 3 entries:

    {[0 1 1]} ⍰ {@sin}
    {[1 0 1]} ⍰ {@cos}
    {[1 1 0]} ⍰ {@tan}
```

```
% можно в качестве ключей использовать массивы
fun = d2({[true true false]})
```

```
fun = 1×1 cell array
    {@tan}
```

```
fun{1}(pi)
```

```
ans = -1.2246e-16
```

Итерирование по коллекциям

Циклы могут перебирать элементы коллекций (но только родных джавовских не могут)

```
% итерирование по ячейкам
A_cell = {1,2,3}
```

A_cell = 1×3 cell

	1	2	3
1	1	2	3

```
for a = A_cell
    class(a)
    disp(a{1})
end
```

```
ans =
'cell'
```

```
     1
ans = 
'cell'
     2
ans = 
'cell'
     3
```

```matlab
% итерирование по структурам
A_struct(3) = struct('f1',3);
A_struct(1).f1 = 1;A_struct(2).f1 = 2;

for st = A_struct
    disp("st(i)=" + st.f1)
end
```

```
st(i)=1
st(i)=2
st(i)=3
```

```matlab
%итерирование по словарям

A_dict = dictionary(["a" "b" "c"],[1 2 3])
```

```
A_dict = 

  dictionary (string ⟼ double) with 3 entries:

    "a" ⟼ 1
    "b" ⟼ 2
    "c" ⟼ 3
```

```matlab
try
    for d = A_dict
        d
    end
catch Ex
    "ss"
end
```

```
ans = 
"ss"
```

```matlab
i = 0;
for key = keys(A_dict)'
    i = i+1
    disp("key => value:  " + key+"=>"+ A_dict(key))
end
```

```
i = 1
key => value:  a=>1
i = 2
key => value:  b=>2
```

```
i = 3
key => value:  c=>3
```

```
A_dict(key)
```

```
ans = 3
```

Не хватает коллекции уникальных элементов типа множество

Вариант 1 методы матлаб для работы с массивами как с множествами

```
clearvars
A = ["c"  "b" "a" "c"];
B = ["a" "d"];
setdiff(A,B) % Элементы множества A не содержащиеся в множестве B
```

```
ans = 1×2 string
"b"          "c"
```

```
setdiff(A,B,'stable') % чтобы сохранить изначальный порядок элементов в массиве
```

```
ans = 1×2 string
"c"          "b"
```

```
intersect(A,B) % Пересечение двух множеств
```

```
ans = 
"a"
```

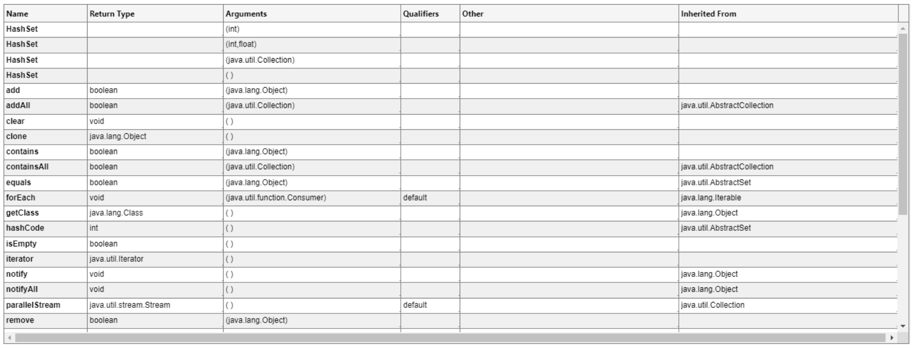
```
unique(A)
```

```
ans = 1×3 string
"a"          "b"          "c"
```

Вариант 2: использовать богатый арсенал java (collections)

Матлаб имеет "встроенный" java 8

```
methodsview(java.util.HashSet) % Графическая оболочка для java документации
```



Name	Return Type	Arguments	Qualifiers	Other	Inherited From
HashSet		(int)			
HashSet		(int,float)			
HashSet		(java.util.Collection)			
HashSet		( )			
add	boolean	(java.lang.Object)			
addAll	boolean	(java.util.Collection)			java.util.AbstractCollection
clear	void	( )			
clone	java.lang.Object	( )			
contains	boolean	(java.lang.Object)			
containsAll	boolean	(java.util.Collection)			java.util.AbstractCollection
equals	boolean	(java.lang.Object)			java.util.AbstractSet
forEach	void	(java.util.function.Consumer)	default		java.lang.Iterable
getClass	java.lang.Class	( )			java.lang.Object
hashCode	int	( )			java.util.AbstractSet
isEmpty	boolean	( )			
iterator	java.util.Iterator	( )			
notify	void	( )			java.lang.Object
notifyAll	void	( )			java.lang.Object
parallelStream	java.util.stream.Stream	( )	default		java.util.Collection
remove	boolean	(java.lang.Object)			

Пример нахождения пересечения двух множеств

```
import java.util.HashSet % java.util.* - импортирует все коллекции из джавы
```

```
jA = HashSet; % вызываем конструктор для джава объекта
jB = HashSet; % вызываем конструктор для джава объекта
methods(jA)
```

```
Methods for class java.util.HashSet:

HashSet          add              addAll           clear            clone            contains         containsAll      equa
```

```

A = ["a" "b" "c"];
B = ["a" "d"];

for iii = A
    add(jA,iii) % добавляем элемент в множество jA
end
```

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

```
for iii = B
    add(jB,iii) % добавляем элемент в множество jB
end
```

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

```
unionAB = clone(jA) % клонирует объект (метод java)
```

```
unionAB =

[a, b, c]
```

```
unionAB.addAll(jB) % функция добавляет элементы множества jB в множество  unionAB
```

```
ans = logical
   1
```

```

jA
```

```
jA =

[a, b, c]
```

```
unionAB
```

```
unionAB =

[a, b, c, d]
```

```

intersectionAB = clone(jA)
intersectionAB =

[a, b, c]
```

```
intersectionAB.retainAll(jB)
```

```
ans = logical
   1
```

```
jA
```

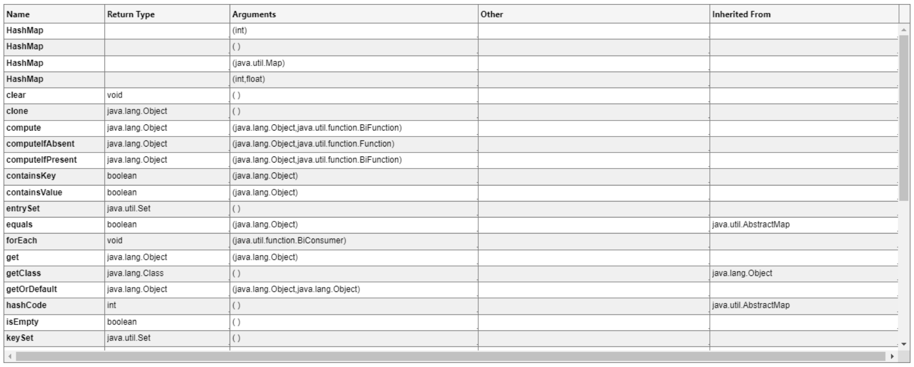
```
jA =

[a, b, c]
```

```
intersectionAB
```

```
intersectionAB =

[a]
```

Вместо типа dictionary можно использовать java.util.HashMap, это возможно будет работать быстрей (скорее всего)

```
methodsview(java.util.HashMap)
```



Name	Return Type	Arguments	Other	Inherited From
HashMap		(int)		
HashMap		( )		
HashMap		(java.util.Map)		
HashMap		(int,float)		
clear	void	( )		
clone	java.lang.Object	( )		
compute	java.lang.Object	(java.lang.Object,java.util.function.BiFunction)		
computeIfAbsent	java.lang.Object	(java.lang.Object,java.util.function.Function)		
computeIfPresent	java.lang.Object	(java.lang.Object,java.util.function.BiFunction)		
containsKey	boolean	(java.lang.Object)		
containsValue	boolean	(java.lang.Object)		
entrySet	java.util.Set	( )		
equals	boolean	(java.lang.Object)		java.util.AbstractMap
forEach	void	(java.util.function.BiConsumer)		
get	java.lang.Object	(java.lang.Object)		
getClass	java.lang.Class	( )		java.lang.Object
getOrDefault	java.lang.Object	(java.lang.Object,java.lang.Object)		
hashCode	int	( )		java.util.AbstractMap
isEmpty	boolean	( )		
keySet	java.util.Set	( )		

```
import java.util.HashMap % можно использовать вместо словарей
jMap = java.util.HashMap; % создается объект java с которым напрямую можно работать 
из матлаб
methods(jMap)
```

```
Methods for class java.util.HashMap:

HashMap             clear               clone               compute             computeIfAbsent     computeIfPresent    contain
```

```
jMap.put("a",figure(1));
jMap.put("b",figure(2));
```

```
keySet(jMap)
```

```
ans =

[a, b]
```

```
jMap.get("b") % вытаксkиваем число
```

```
ans = 
  Figure (2) with properties:

      Number: 2
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [488 242 560 420]
       Units: 'pixels'

  Show all properties
```

```
jMap
```

```
jMap =

{a=matlab.ui.Figure, b=matlab.ui.Figure}
```

```
jArrayObj = jMap.values().toArray()
```

```
jArrayObj =

  java.lang.Object[]:

    [matlab_ui_FigureBeanAdapter0]
    [matlab_ui_FigureBeanAdapter0]
```

```
methods(jArrayObj)
```

```
Methods for class java.lang.Object[]:

equals      getClass    hashCode    notify      notifyAll   toString    wait
```

```
f_array= arrayfun(@(i)jArrayObj(i), 1:jMap.size(),"UniformOutput",false);
f_array{1}
```

```
ans = 
  Figure (1) with properties:

      Number: 1
        Name: ''
       Color: [1 1 1]
    Position: [488 242 560 420]
       Units: 'pixels'

  Show all properties
```

```
function return_value = like_example(be_like_me)
    return_value = zeros(numel(be_like_me),'like',be_like_me);
    return_value = return_value*be_like_me(:);
end
function A = fill_by_row()
    N = 5000;
    A = zeros(N);
    for iii=1:N % внешний цикл перебирает строки
        for jjj=1:N
            A(iii,jjj) = 5;
        end
    end
end
function A = fill_by_column()
    N = 5000;
    A = zeros(N);
    for jjj=1:N % внешний цикл перебирает колонки
        for iii=1:N
            A(iii,jjj) = 5;
        end
    end
end
function A=fill_by_column_no_memalloc()
    N = 5000;
    for jjj=1:N % внешний цикл перебирает колонки
        for iii=1:N
            A(iii,jjj) = 5;
        end
    end
end
function MAT=fill_by_column_reverse_order()
    N = 5000;
    for jjj=N:-1:1 % внешний цикл перебирает колонки
        for iii=N:-1:1
            MAT(iii,jjj) = 5;
        end
    end
end

function [r_str,r_ch] = gen_random_string(N)
        alfabeth = 'a':'y';
        n = numel(alfabeth);
        rand_inds = randi(n,[1,N]);
        r_ch = alfabeth(rand_inds);
        r_str = string(r_ch);
end

%% Сравнение операций, выполняемых непосредственно для всей матрицы и перебором элементов матрицы
function A = sin_in_circle(A)
    N = size(A);
    for jjj=1:N(2) % внешний цикл перебирает колонки
        for iii=1:N(1)
            A(iii,jjj) = sin(A(iii,jjj));
        end
    end
```

```
end
function A = sin_direct(A)
    A = sin(A);
end
function A = sin_in_circle_line_index(A)
    N = numel(A);
    for iii=1:N
        A(iii) = sin(A(iii));
    end
end
%
function out = ALL(A)
    out = sum(A,'all');
end
% что быстрей итерирование по коллекции или итерирование с индексацией
function s = indexwise_iter() % индексирование по индексам
    A = rand(100000,1);
    s=0;
    for iii = 1:numel(A)
        s = s+ A(iii);
    end
end
function s = elementwise_iter()
    A = rand(100000,1);
    s=0;
    for a = transpose(A)
        s = s+ a;
    end
end
% Пример исопльзования структур типа cell  - функция с произвольным числом
% аргументов
function varar_fun(varargin)
    counter = 0;
    for arg = varargin
        counter = counter + 1;
        disp("arg" + counter);
        disp(arg{1})
    end
end

function folder = get_folder()
% текущая папка
folder = fileparts(matlab.desktop.editor.getActiveFilename);
end
```