Данные в MATLAB. Структуры.

Данные в системе MATLAB группируются по классам. Всего в MATLAB имеется 14 базовых классов (или типов) данных. Каждый из этих типов является формой массива (array). Этот массив может иметь минимальный размер 0×0 или может иметь произвольную размерность по любой координате. Все 14 типов приведены на рис. 1.

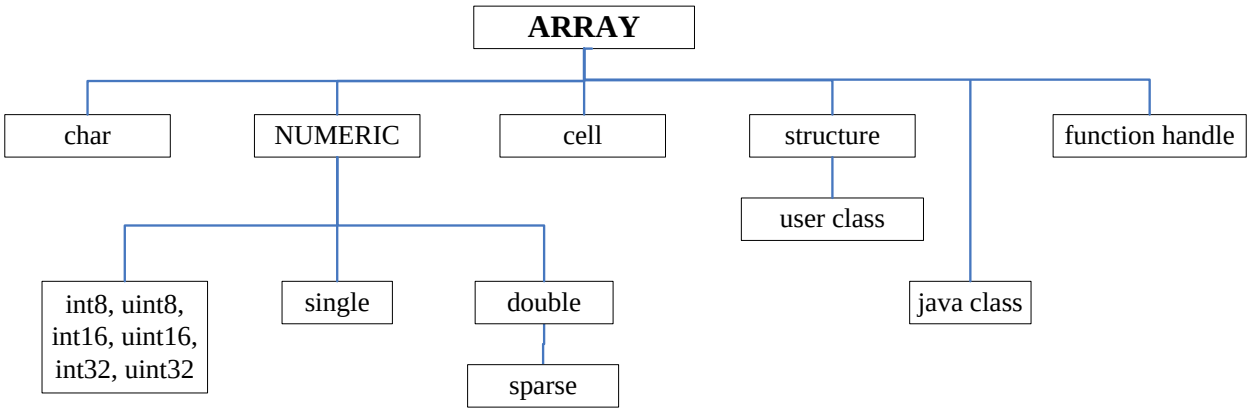


Рис. 1.

Наиболее используемым является класс данных *numeric*. Он содержит целые и вещественные числа. Целые числа со знаком и без знака задаются различными функциями, в зависимости от необходимого размера памяти в байтах, выделяемой для чисел. Вещественные данные могут быть числами одинарной (4 байта) и двойной (8 байт) точности, задаваемыми функциями *single* и *double* соответственно. Все числа при выполнении арифметических операций должны быть числами двойной точности. Перевод целых чисел в класс *double* осуществляется функцией **`double`**. Система выполняет все вычисления в арифметике с плавающей точкой, в отличие от таких систем как Maple и Mathematica, где преобладает целочисленное представление и символьная обработка данных.

Основные типы:

double – числовые массивы с числами двойной точности (8 байт);

char – строчные массивы с элементами-символами;

sparse – разреженные матрицы с элементами-числами двойной точности;

cell – массивы ячеек;

struct – массивы записей с полями, которые также могут содержать массивы;

Представление массивов в MATLAB

Пакет MATLAB создавался как инструмент для решения задач линейной алгебры. За основу представления данных разработчиками были взяты матрицы (массивы), поэтому любая скалярная переменная x рассматривается как матрица размерности 1×1. Описывать скалярную переменную как массив и индексировать нет необходимости. Под массивом понимается упорядоченная, пронумерованная совокупность данных. Различаются массивы по числу измерений: одномерные, двумерные, многомерные. Одномерный массив в MATLAB представляет собой вектор-столбец или вектор-строку, а двумерные – матрицей. Массивы должны иметь имя.

Массивы могут быть не только числовыми, но и строковыми. Строковая переменная определяется как последовательность символов, заключенных в апострофы и относится к классу *char*. В массивах строковых переменных число символов в каждом элементе массива должно быть одинаковым. Дополнить недостающие символы в каком-либо элементе можно пробелами.

Операции над строками

Функция	Описание	Пример
`char`	Преобразование целого числа в символ, используя кодировку ASCII, или объединение символов в массив	• Объединение строк: `>> both = char(msg1,msg2)` `both =` `There are 39.3701 inches in a meter` `There are 61.02 cubic inches in a liter`

		• Обращение к элементу по кодировке ASCII: `>> char(49)` `ans =` `1` `>> char([77 65 84 76 65 66])` `ans =` `MATLAB`
strcat	Горизонтальное объединение строк	
strvcat	Вертикальное объединение строк	
strcmp	Сравнить строки	`>> msg1 =` `['There are ',num2str(100/2.54),'` `inches in a meter'];` `>> msg2 = sprintf('There are %5.2f` `cubic inches in a liter',1000/2.54^3);` `>> strcmp(msg1,msg2)` `ans =` `0`
strncmp	Сравнить n символов строк	***Пример 1***: `>> strncmp(msg1,msg2,9)` `ans =` `1` ***Пример 2*** *(проверка утвердительного ответа на вопрос «продолжить?»):* `input('Do you want to continue (y or n) ? ','s');` `if strcmp(lower(ans(1)),'y')` `go_ahead` `else` `return` `end`
findstr	Найти заданную строку в составе другой строки	`>> findstr('in',msg1)` `ans =` `19 26` `>> msg1(19:20)` `ans =` `in`
strjust	Выравнивание массива символов	
strmatch	Найти все совпадения	
strrep	Заменить одну строку другой	`>> str = 'I go now';` `>> strrep(str,'go','eat snails')` `ans =` `I eat snails now`
strtok	Найти часть строки, ограниченную разделителями	
upper	Перевести все символы строки в верхний регистр	
lower	Перевести все символы строки в нижний регистр	

Преобразования строк

num2str	Преобразование числа в строку **Синтаксис:** *stringValue* = num2str(*numericValue*)	***Пример 1:*** `>>msg1=['There are ',num2str(100/2.54), ' inches in a meter']` `msg1 =` `There are 39.3701 inches in a meter` `>> x = sqrt(2);` `>> disp(['x = ',num2str(x)]);`

		x = 1.4142 ***Пример 2:*** >> A = eye(3); S = num2str(A); B = str2num(S); >> A-S ??? Error using ==> - Matrix dimensions must agree. >> A-B ans = 0 0 0 0 0 0 0 0 0 Хотя A и S содержат одинаковые значения, они не эквивалентны. A – числовая матрица, в то время как S – матрица символов.
str2num	Преобразование строки в арифметическое выражение и его вычисление	
int2str	Преобразование целого в строку	>> for i = 1:3 disp(['Doing loop number ' int2str(i) ' of 3']) end Doing loop number 1 of 3 Doing loop number 2 of 3 Doing loop number 3 of 3
mat2str	Преобразование матрицы в строку	
str2mat	Объединение строк в матрицу	
sprintf	Запись форматированных данных в виде строки	>> msg2 = sprintf('There are %5.2f cubic inches in a liter',1000/2.54^3) msg2 = There are 61.02 cubic inches in a liter
fprintf	**Syntax:** fprintf(fileID,*outFormat*,*outVars*) Здесь *outFormat* – строка, с указанием формата, которая конвертирует результат *outVars* в строки, которые выводятся. Если fileID отсутствует, результат выводится в *Command window*. При его указании результат записывается в соответствующий файл.	>> x = 3; >> fprintf('Square root of %g is %8.6f\n',x,sqrt(x)); The square root of 3 is 1.732051
sscanf	Прочитать строку с учетом формата	

Замечания:

Зам. 1: конструкция

```
disp(['x = ',num2str(x)]);
```

работает в случае, когда x – вектор-строка, и не работает в случае, когда x – вектор-столбец или матрица:

```
>> y = [1 2];
>> z = y';
>> disp(['z = ',num2str(z)])
??? Error using ==> horzcat
All matrices on a row in the bracketed expression must have the same number
of rows.
```

Для вывода вектор-строки или матрицы следует использовать две команды `disp`:

```
>> disp('z = '); disp(z)
z =
1
2
```

Зам. 2: второй входной параметр в функции `num2str` определяет формат вывода данных, аналогичный языку программирования С. Спецификация формата задается знаком «%», последующим указанием ширины и символ преобразования: d, f, e и т.д. (см. таблицу ниже). Основная идея заключается в использовании строки символов, начинающейся со знака «%», чтобы контролировать форматирование. Например, для вывода пять знаков после запятой в поле 12 символов в экспоненциальной форме, надо будет записать следующее:

```
>> num2str(pi,'%12.5e')
ans =
3.14159e+00
>> num2str(-pi,'%12.5e')
ans =
-3.14159e+00
>> num2str(pi*1e100,'%12.5e')
ans =
3.14159e+100
```

Можно также добавить при этом дополнительный текст, например:

```
>> num2str(pi,'Pi has a value of %12.5e, or thereabouts.')
ans =
Pi has a value of 3.14159e+00, or thereabouts.
```

Следующая таблица взята из документации по MATLAB.

Спецификатор	Форма вывода
%c	символ
%d	десятичное число со знаком
%e	экспоненциальная форма
%f	вещественное число с плавающей точкой
%g	Более компактная форма %e или %f. Незначащие нули не выводятся.
%i	десятичное число со знаком
%o	Восьмеричная запись (без знака)
%s	строка символов
%u	Десятичная запись (без знака)
%x	шестнадцатиричное число

Массивы ячеек

Если рассматриваются массивы ячеек (*cell array*), то обращение к элементу такого массива осуществляется посредством фигурных скобок. Элементами такого массива могут быть: числа, строки (различной длины), массивы, другие массивы ячеек и т.д.

`c{i}` – обращение к *i*-му элементу одномерного массива ячеек;

`c{i,j}` – обращение к элементу двумерного массива ячеек, стоящего на пересечении *i*-ой строки и *j*-го столбца.

Пример:

```
t = {'O sacred receptacle of my joys,';
'Sweet cell of virtue and nobility,';
'How many sons of mine hast thou in store,';
'That thou wilt never render to me more!'}
```

Т.о. мы создали массив ячеек 4 x 1:

```
whos
```

```
  Name    Size     Bytes  Class
  t       4x1      530    cell array

>> t(1)
ans =
'O sacred receptacle of my joys,'
>> t{1}
ans =
O sacred receptacle of my joys,
>> t{1}(1)
ans =
O
>> t{1}(1:8)
ans =
O sacred
```

Добавим в массив еще один элемент – матрицу 3 x 3 в первую строку второго столбца:

```
>> t{1,2} = spiral(3)
t =
[1x31 char] [3x3 double]
[1x34 char] []
[1x41 char] []
[1x39 char] []
```

MATLAB заполнил остальные ячейки во втором столбце пустыми ячейками. Для обращения к определенной ячейке мы использовали фигурные скобки: `t{1,2}`. Если бы мы использовали круглые скобки, программа выдала бы ошибку:

```
>> t(1,2) = spiral(3)
??? Conversion to cell from double is not possible.
```

Это происходит потому, что существует разница между индексированием *ячеек* и индексированием их *содержимого*. Например, чтобы извлечь слово “virtue” из второй строки цитаты в первом столбце, необходимо обратиться к *ячейке* {2,1}, а потом получить значения от 15 до 20 из *содержимого* этой ячейки:

```
>> t{2,1}(15:20)
ans =
virtue
```

Массив ячеек может содержать в себе другие массивы ячеек. Например:

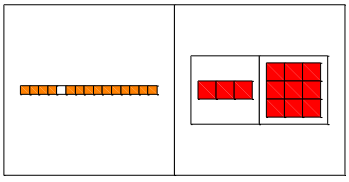
```
>> t = {'Fred Flintstone' {[1 2 3], spiral(3)}}
t =
'Fred Flintstone' {1x2 cell}
```

По умолчанию MATLAB выводит массив ячеек в краткой форме, как в приведенных выше примерах. Можно отобразить также информацию, используя функцию **`celldisp`**:

```
>> celldisp(t)
t{1} =
Fred Flintstone
t{2}{1} =
1 2 3
t{2}{2} =
7 8 9
6 1 2
5 4 3
```

Или же, можно получить графическое отображение информации с помощью функции **`cellplot`**:

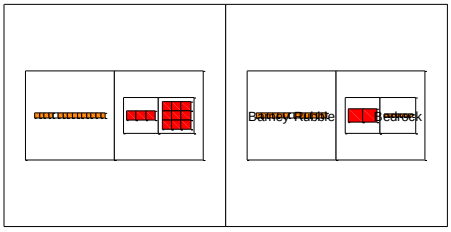
```
>>cellplot(t)
```



Левый квадрат – первая ячейка, содержащая строку 'Fred Flintstone'. Правый квадрат – вторая ячейка, содержащая массив ячеек 1 × 2, куда, соответственно, входят вектор [1 2 3] и матрица-спираль (3).

Для индексирования вложенных массивов ячеек следует использовать фигурные скобки {} для достижения требуемого уровня вложенности, а затем использовать круглые скобки () для доступа к их содержимому. Например:

```
>> tt = {t {'Barney Rubble' {[-1 1] , 'Bedrock'}}}
tt =
{1x2 cell}
{1x2 cell}
>> cellplot(tt)
```



```
>> tt{2}{2}
ans =
[1x2 double] 'Bedrock'
>> tt{2}{2}{1}
ans =
-1 1
>> tt{2}{2}{1}{2}
??? Cell contents reference from a non-cell array object.
>> tt{2}{2}{1}(2)
ans =
1
```

Пример:

```
a={'Символьная строка', 27, [5 7]; {1},{[ 2 5 3]},{'строка в ячейке'}}
a =
    'Символьная строка'    [      27]    [1x2 double]
             {1x1 cell}    {1x1 cell}    {1x1 cell  }

% Чтение и запись отдельных элементов (использование фигурных скобок):
>> a{1,3}
ans =
     5     7
>> a{1,3}=[10,12,27]
>> a{1,3}
ans =
    10    12    27
a =
    'Символьная строка'    [      27]    [1x3 double]
             {1x1 cell}    {1x1 cell}    {1x1 cell  }

% Результаты обращения к элементам массива ячеек: () и {}
>> a(1,3)
ans =
    [1x3 double]
>> a{1,3}
ans =
    10    12    27
```

Команды **`size`** и **`length`** в случае с массивами ячеек ведут себя так же, как и с массивами чисел:

```
>>a={1,2,3}
a =     [1]    [2]    [3]
>>size(a)
ans =     1     3
```

```
>>length(a)
ans =     3
```

Взаимное преобразование массивов double и cell

Используются команды:

num2cell – преобразование числового массива в массив ячеек

cell2mat – преобразование массива ячеек в числовой массив (обратная команде **num2cell**).

Рассмотрим массив A:

```
A=[1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
```

Вызов команды **num2cell** с одним аргументом

```
num2cell(A)
ans =
    [1]    [2]    [3]
    [4]    [5]    [6]
```

Результат – массив ячеек.

Вызов команды **num2cell** с двумя аргументами

При вызове с двумя аргументами первый содержит исходный массив, а второй – номер измерения, по которому матрицу предстоит «разрезать» на полосы:

```
%num2cell(A,1) «режет» матрицу по вертикали (по столбцам)
n1=num2cell(A,1)
n1 =
    [2x1 double]    [2x1 double]    [2x1 double]
n1{2}
ans =
     2
     5

% num2cell(a,2) «режет» матрицу по горизонтали (по строкам)
n2=num2cell(A,2)
n2 =
    [1x3 double]
    [1x3 double]
n2{1}
ans =
     1     2     3
```

Действие команды **cell2mat** обратно действию **num2cell**. Вызов осуществляется только с одним входным аргументом:

```
cell2mat(num2cell(A))
ans =
     1     2     3
     4     5     6
```

Результат – числовая матрица A

Команда **mat2cell** – разбиение матрицы на массив ячеек. На входе функция принимает три аргумента – исходный массив и два массива, содержащих размеры вертикальных и горизонтальных блоков.

Пример:

Рассмотрим матрицу размерности 5x7. Разобьем матрицу на блоки линиями: по вертикали (горизонтальные линии: 2 строки, 1 строка и 2 строки), по горизонтали (вертикальные линии: 2 столбца, 3 столбца, 2 столбца).
В соответствии с этим получим два массива: [2 1 2] и [2 3 2].

```
X=[1 2 3 4 5 6 7;8 9 10 11 12 13 14;15 16 17 18 19 20 21; 22 23 24 25 26 27
28;...
29 30 31 32 33 34 35]
X =
     1     2     3     4     5     6     7
     8     9    10    11    12    13    14
    15    16    17    18    19    20    21
    22    23    24    25    26    27    28
    29    30    31    32    33    34    35

Y=mat2cell(X,[2 1 2],[2 3 2])
Y =
    [2x2 double]    [2x3 double]    [2x2 double]
    [1x2 double]    [1x3 double]    [1x2 double]
    [2x2 double]    [2x3 double]    [2x2 double]

Y{3,2}
ans =
    24    25    26
    31    32    33

Y{3}
ans =
    22    23
    29    30
```

Структуры

Структуры относятся к сложным типам. Они могут содержать разнородные данные, относящиеся к некоторому именованному объекту. Поля структуры оделяются друг от друга точкой. Обращение к полям структуры осуществляется при помощи оператора «точка». Рассмотрим пример:

```
>> staff.name = 'John Smith'
staff =
name: 'John Smith'
>> staff.age = 43
staff =
name: 'John Smith'
age: 43
>> staff.favourites = [1 42 37]
staff =
name: 'John Smith'
age: 43
favourites: [1 42 37]
```

Т.о. мы создали структуру «staff» размерности 1 × 1:

```
>> whos
Name   Size  Bytes   Class
staff  1x1   424     struct array
```

Данная структура имеет три поля: name, age, и favourites:

```
>> staff
staff =
```

```
    name: 'John Smith'
    age: 43
    favourites: [1 42 37]
```

Для добавления в эту структуру данных о еще одном человеке, добавляется номер в круглых скобках, тем самым указывая на новый элемент структуры:

```
staff(2).name = 'Jane Smith';
staff(2).age = 30;
staff(2).favourites = [pi eps realmax realmin NaN Inf];
```

Размеры полей могут отличаться для каждого элемента структуры. Например, вектор favourite у элемента Jane Smith длиннее, чем соответствующий вектор у John Smith.

Структура в пакете MATLAB создается функцией **`struct`**, которая в качестве входных параметров принимает набор пар – имени и значения поля:

`struct (имя1, значение1, имя2, значение2, …..).`

Пример:

Создание структуры с помощью функции **`struct`**

```
s=struct('name', 'Sherlock Holmes','group','001')
```

Создание массива структур простым добавлением. Введем 2-й элемент массива структур

```
s(2)=struct('name','James Bond','group','007')
```

Доступ к полю элемента массива структур

```
s(2).name
```

Добавление поля к структуре

```
s(2).rating=[5 4 5 4 5 4]
s(2)
```

Внимание!

При объявлении структуры, у которой одно из полей является массивом ячеек, используются одни «лишние» скобки. Если их не поставить, то будет создан массив структур. При этом функция **`struct`** работает следующим образом.

- Если лишь одно поле структуры в качестве значения имеет массив ячеек, то создается массив структур, в котором каждый из элементов принимает значения из массива ячеек, а остальные поля фиксированы.
- Если два или более полей заданы массивами ячеек, то эти массивы должны быть одинаковой длины и функция **`struct`** создает массив структур, в котором элементы поочередно принимают соответствующие значения.
- Если же количество ячеек разное, то выдается сообщение об ошибке

Структура с одним полем ячеек (вывод всей информации):

```
d=struct('name','Harry Potter','subject',{{'Transfiguration' 'Portions'
'Defense Against the Dark Arts'}},'scores',[2 2 5])
d =
       name: 'Harry Potter'
    subject: {'Transfiguration'  'Portions'  'Defense Against the Dark
Arts'}
     scores: [2 2 5]
```

Структура с массивом ячеек (вывод «по элементам массива ячеек»). Одно поле «меняется», другие «закреплены»:

```
d=struct('name', 'Harry Potter', 'subject',{ 'Transfiguration' 'Portions'
'Defense Against the Dark Arts'},'scores',[2 2 5])
d(1)
d(2)
d(3)
```

```
d =
1x3 struct array with fields:
    name
    subject
    scores
ans =
       name: 'Harry Potter'
    subject: 'Transfiguration'
     scores: [2 2 5]
ans =
       name: 'Harry Potter'
    subject: 'Portions'
     scores: [2 2 5]
ans =
       name: 'Harry Potter'
    subject: 'Defense Against the Dark Arts'
     scores: [2 2 5]
```

Структура с массивами ячеек (элементы массивов ячеек должны быть согласованы). Не ячейки закреплены, массивы ячеек согласованы:

```
d=struct('name', 'Harry Potter', 'subject',{ 'Transfiguration',...
'Portions', 'Defense Against the Dark Arts'},'scores',{2 2 5})
d(1)
d(2)
d(3)
d =
1x3 struct array with fields:
    name
    subject
    scores
ans =
       name: 'Harry Potter'
    subject: 'Transfiguration'
     scores: 2
ans =
       name: 'Harry Potter'
    subject: 'Portions'
     scores: 2
ans =
       name: 'Harry Potter'
    subject: 'Defense Against the Dark Arts'
     scores: 5
```

Заполнение «базы данных». Согласованный вывод элементов массива ячеек:

```
bd=struct('Name', {'Harry Potter' 'Hermione Granger' 'Ronald Weasley'},...
'Sum_result',{83,79,90},'Rating',{2 1 3})
bd(1)
bd(2)
bd(3)
bd =
1x3 struct array with fields:
    Name
    Sum_result
    Rating
```

```
ans =
           Name: 'Harry Potter'
     Sum_result: 83
         Rating: 2
ans =
           Name: 'Hermione Granger'
     Sum_result: 79
         Rating: 1
ans =
           Name: 'Ronald Weasley'
     Sum_result: 90
         Rating: 3
```

Добавление в «базу данных»

```
bd(4)=struct('Name','Draco Malfoy','Sum_result','95','Rating','1')
```

Для работы со структурами имеются следующие функции: **isfield**, **getfield**, **setfield**, **rmfield**, **fieldnames**.

Функция	Действие
isfield(s,fldname)	возвращает 1 если у структуры s есть поле fldname и 0 в противном случае.
getfield (s,fldname)	возвращает значение поля fldname структуры s, т.е. является синонимом для s.fldname.
setfield (s,fldname,fldvalue)	возвращает копию объекта s, у которой полю fldname присвоено значение fldvalue. Сам объект s при этом не изменяется, т.е. это неравносильно присваиванию s.fldname=fldvalue.
rmfield(s,fldname)	предназначена для удаления поля fldname у объекта s. Для того чтобы добавить новое поле, достаточно просто присвоить ему значение: s.new_field_name=some_value.
fieldnames(s)	возвращает массив ячеек, содержащий имена полей структуры s.

Примеры использования этих команд.

```
isfield(bd,'name')
getfield(bd,'Name')
getfield(bd(3),'Name')
new_record=setfield(bd(3),'Name', 'Ronald Weasley')
bd(3)
new_record
```

Добавление нового поля и присвоение в 1-записи структуры полю group значения 112

```
bd(1).group='007'
fieldnames(bd)
```

Удаление поля group из структуры bd (обратить внимание, слева также стоит структура!):

```
bd=rmfield(bd,'group')
fieldnames(bd)
```

Групповая операция копирования элементов поля структуры в массив (обратить внимание на применение квадратных скобок!):

```
r=[bd.Rating]
```