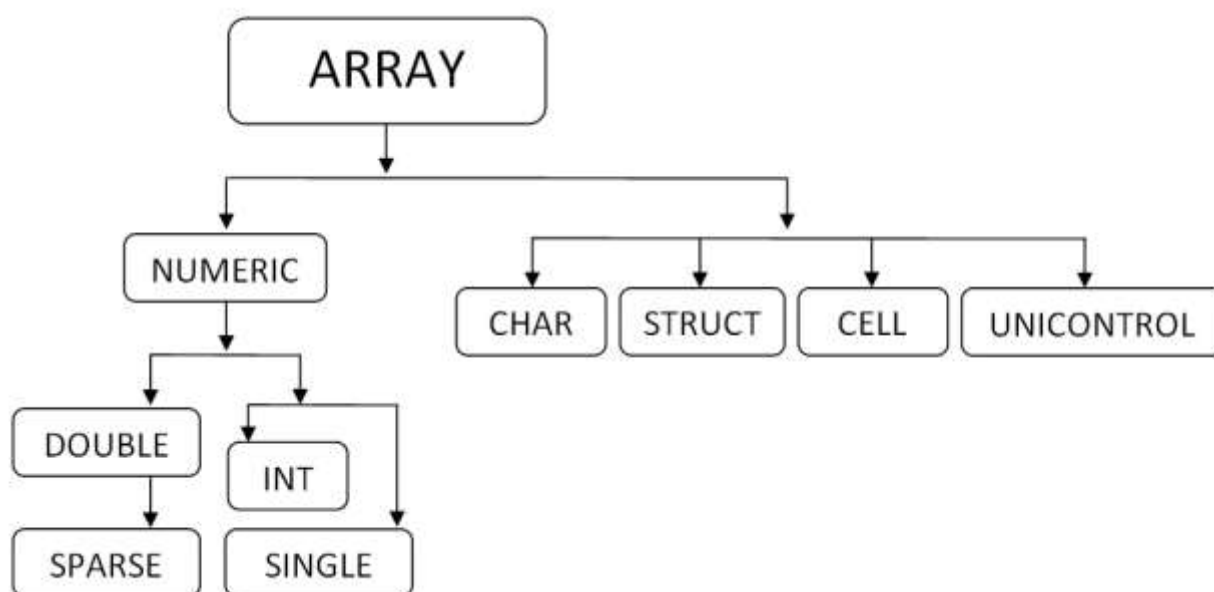


Основные типы данных

Любой объект в ML, в том числе скаляр, является массивом. Класс `ARRAY` – законодатель класса массивов, в котором разработаны методы функционирования всех дочерних объектов.

Хранение массивов в ML осуществляется в векторной форме последовательно по столбцам, поэтому поддерживается как двойная (матричная) нумерация, так одинарная (векторная). Основой операций с массивами является согласование размерностей. Заметим, что класс `ARRAY` является родительским для всех потомков и определяет все объекты более низкого уровня как массивы (матрицы) так и правила (методы).

На рисунке представлена схема иерархии классов (типов данных) в ML.



Методы класса `Array`

`ndims` – возвращает количество размерностей многомерного массива (любой природы), `n = ndims(a)`;

`size` – определяет вектор `v = size(a)` размерностей массива;

так, для `n = 2`, `[m, p] = size(a)`;

`length(a)` – определяет длину вдоль большей размерности;
`length(a(:))` определяет длину массива, записанного вектором;
`numel(a)` – количество элементов массива;
`disp(display)` – визуализация объектов, не подавляется знаком (;)

Типы данных Numeric и Double

Все объекты в ML делятся на числовые, если `isnumeric(a) = 1` равно логической единице, и нечисловые, если `isnumeric = 0`.

В момент создания числового объекта идентифицируется класс потомка Numeric, которому он принадлежит. ML ориентирован на матричные вычисления двойной точности с элементами класса Double.

Пример 1. Логическая единица не является числовым объектом

```
% Справка по элементарным функциям
clear
a=rand(2,3);
% генерирование матрицы из двух строк, трех столбцов,
элементы которой равномерно распределены на отрезке
[0,1]
b=isnumeric(a)
% результат: 1 , т.е. a – числовой объект
islogical(b)
% результат: 1, т.е. b – логический объект
isnumeric(ans)
% результат: 0, т.е. логический объект – нечисловой
```

В силу свойств наследования имеем:

- скаляр – тоже массив, минимальный элемент размера (1,1);
- в памяти матрица хранится как вектор, записанный последовательно по столбцам и поддерживается одинарная и двойная нумерация;

Заметим, что диапазон вещественных чисел `[realmin, realmax]`, здесь `realmin`, `realmax` системные переменные минимальное и максимальное вещественные значения.

Способы создания объектов Double

Пример 2. Задание объектов перечислением:

```
% вектор-столбец
a = [1; 2; 3];
% задание вектор-строки
a = [1 2 3]
% задание вектор-строки, иной синтаксис
a = [1,2,3]
% задание вектора с элементами арифметической прогрессии (первый член, шаг, последний)
a = [a1 : aStep:aEnd]
```

Пример 3. Задание объектов с помощью специальных матриц:

```
a = rand(size(b)) % - матрица с элементами, полученными генератором случайных чисел, равномерно распределенных на отрезке [0,1], такого же размера, как некоторая матрица b

a = randn(m, n) % - размер генерируемой матрицы m на n; используется генератор нормального распределения с нулевым средним и единичной дисперсией.

a = ones(n) % - создается квадратная матрица из единиц размера n
b = eye(n) % - единичная матрица
a = zeros(n) % - нулевая матрица
```

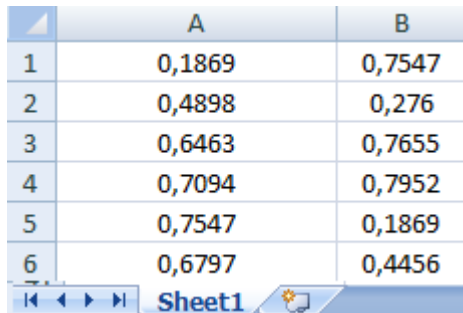
Задание объектов импортированием можно производить с помощью непосредственного импортирования и чтением из внешних файлов:

- команды `File → ImportData`,
- чтение экселевского файла:

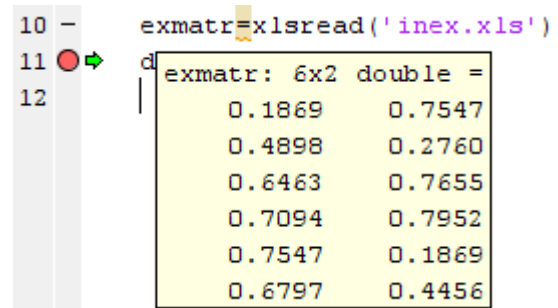
```
namematrix=xlsread(pathname.filename) , здесь pathname.filename – строка (путь и имя файла);
```

Заметим, что имена листов следует писать на английском языке, файлы не должны нарушать структуру матриц; создается файл по имени выходной переменной `namematrix`.

Ниже, на рисунке, приведен пример чтения данных из файла Excel и результат чтения в ML:



	A	B
1	0,1869	0,7547
2	0,4898	0,276
3	0,6463	0,7655
4	0,7094	0,7952
5	0,7547	0,1869
6	0,6797	0,4456



```
10 - exmatr=xlsread('inex.xls')
11 →
12
```

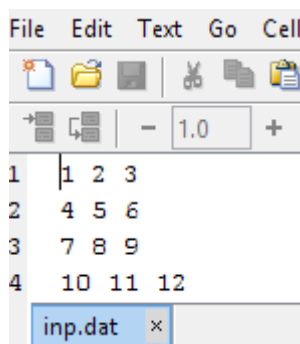
```
exmatr: 6x2 double =
    0.1869    0.7547
    0.4898    0.2760
    0.6463    0.7655
    0.7094    0.7952
    0.7547    0.1869
    0.6797    0.4456
```

Файл `inex.xls`

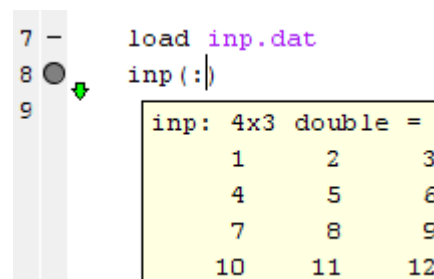
Имя листа `Sheet1`

- импортрование текстового файла

`load filename` (текстовый файл, см. пример, пусть `filename` это `inp.dat`)



	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9
4	10	11	12



```
7 - load inp.dat
8 → inp(:)
9
```

```
inp: 4x3 double =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

- форматированный ввод

сначала открывается файл `pathname.filename`, записанный строкой в качестве первого аргумента:

```
fid = fopen('pathname.filename', 'permission')
```

второй аргумент, `'permission'` – указывает на форму доступа (`'r'` – чтение из файла; `'w'` – запись в новый файл, `'a'` – добавление в существующий файл, подробнее см. `help`); `fid` – файловый идентификатор, который система связывает с файлом (здесь подразумевается, что файл расположен в рабочей папке) или указывает на ошибку (например, отрицательное значение `fid`).

Затем происходит считывание данных из файла:

```
a = fscanf(fid, format, size),
```

здесь `format = [' %g %f %e %s %c %d \n']`,

указывает, что в каждой строке файла записано шесть чисел, указанного формата:

`%g` – с плавающей точкой машинного представления

`%f` – с фиксированной точкой

`%e` – с плавающей точкой

`%s` – строка, пробелы в которой не учитываются

`%c` – строка, учитываются пробелы

`%d` – целые десятичные

символ `\n` – переход считывания на следующую строку (перевод каретки);

`size` – размер массива (матрицы), обусловлен машинным представлением, т.е. колонки матрицы, последовательно записаны в вектор-строку

`size = [m, n]` `m` – количество элементов в строке; `n` – количество элементов в столбце, `n = inf`, если неизвестно количество строк в файле (длина столбца), из которого производится чтение, т.е. `size = [m, inf]` (см. далее рисунок с примерами, случай `size = [2, inf]`).

Обратите внимание, что после считывания получаем матрицу, транспонированную по отношению к заданной в файле, но операция транспонирования `A.'` поставит все на свои места.

```
clear
fid1=fopen('inpnum.txt','r')
A=fscanf(fid1,'n=%d eps=%g x0=%f \n', [3,4])
A
```

A: 3x4 double =

1.0000	2.0000	3.0000	4.0000
0.2000	0.0500	0.0001	0.0000
3.2564	3.1419	3.1417	3.1415

	n	eps	x0
1	n=1	eps=2.0e-01	x0=3.2564
2	n=2	eps=5.0e-02	x0=3.1419
3	n=3	eps=1.0e-04	x0=3.1417
4	n=4	eps=1.0e-05	x0=3.1415

inpnum.txt

```
clear
fid=fopen('inpinf.txt','r')
A=fscanf(fid,'%d %g \n', [2,inf])
A
```

A: 2x4 double =

1.0000	2.0000	3.0000	4.0000
0.2000	0.0500	0.0001	0.0000

1	1	2.0e-01
2	2	5.0e-02
3	3	1.0e-04
4	4	1.0e-05

inpinf.txt

Операцию транспонирования A' необходимо применить после считывания в обоих примерах, это обусловлено машинным представлением – хранением матриц вектором, записанным по столбцам.

- форматированный вывод

```
fprintf(fid, format, namevar)
```

Запись в файл `namevar` (записанный строкой с необходимым расширением); `fid` – файловый идентификатор файла, который уже открыт и пустой или ранее создан;

`format` – строка по аналогии с форматированным вводом, здесь далее приведен пример формата, который еще имеет поясняющие надписи

```
['points number= %d   x= %g   y=%e   surf=%f \n' ]
```

`fclose(fid)` – команда, закрывающая созданный файл.

Пример 4. Создание шахматной структуры, без использования циклов

```
%% шахматный порядок
figure
%m - нечетное; n-любое
m=9
n=6
A=rand(m,n)
A(1:2:end)=0

mytitle=['m=',num2str(m),' - число строк, нечетное']
subplot(2,2,1),spy(A),title(mytitle)

m=8 %m - четное;
n=6
A=rand(m,n)
A(1:2:end)=0
mytitle=['m=',num2str(m),' - четное']
subplot(2,2,2),spy(A),title(mytitle)
A=rand(m,n)
AA=A
A=[rand(1,n);A]
```

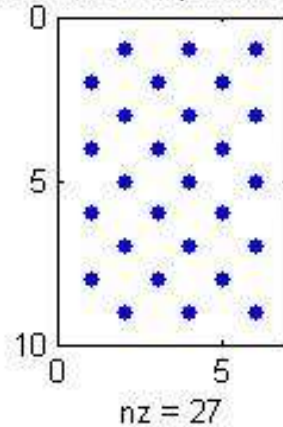
```
A(2:2:end)=0
```

```
A(1,:)=[]
```

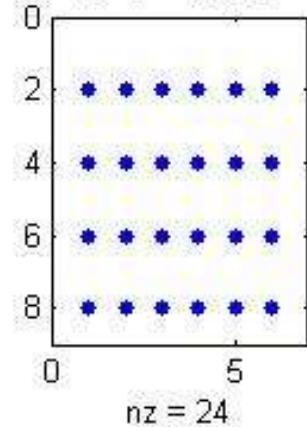
```
mytitle=['добавляем 1-ю строку; тот же алгоритм, и уда-  
ляем 1-ю строку']
```

```
subplot(2,2,3),spy(A), title(mytitle)
```

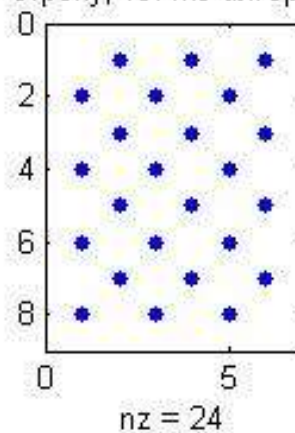
m=9 - число строк, нечетное



m=8 - четное



добавляем 1-ю строку; тот же алгоритм, и удаляем 1-ю строку



Задания для самостоятельного решения

Задание 1

- Построить блочно-диагональную матрицу, которая состоит из n блоков (n – целое, генерируется случайным образом на отрезке $[4, 8]$ с помощью `randi` или `randint` – зависит от версии MatLab).
- Блоки строятся генератором равномерно распределенных чисел на отрезке $[0, 1]$, размер каждого из n блоков определяется арифметической прогрессией $n : 1 : 2n-1$.
- Вывести на экран структуру матрицы командой `spy`.

Задание 2

- Построить матрицу A пятого порядка с помощью генератора `rand`.
- Вычислить $A' A$ и $A' + A$ и доказать, что полученные матрицы симметричны. При выполнении задания циклы использовать нельзя.

Задание 3

- Второй и предпоследний блоки блочно-диагональной матрицы (см. задание 1) определяют подматрицу блочной матрицы, начинающейся с $n+1$ –й строки и столбца и до $end - (2n-1)$ строки и столбца.
- Требуется передвинуть выбранные блоки так, чтобы они разместились в вершинах побочной диагонали подматрицы.
- Структуру матрицы отобразить на экране с помощью команды `spy`.

Задание 4

- Постройте заполненную матрицу размера (m, n) .
- Расставьте нули в матрице в шахматном порядке, не используя операторов цикла.
- Рассмотрите случаи для четного и нечетного значений m .
- Отобразите на экране структуру исходной матрицы и результата в смежных осях. Для этого используйте команды `spy` и `subplot`.