**Table of Contents**



# Семинар 1. Базовые типы МАТЛАБ

- Типы double, logical
- Массивы, размерность, индексирование

### Тип double - основной числовой тип

Число с плавающей точкой x , хранится в виде:

$$x = (-1)^s \cdot (1 + f) \cdot 2^e$$

где:

- *s* определяет знак
- *f* - мантисса, для которой $0 \le f < 1$.
- *e* - экспонента.

*s*, *f*, и *e* определяются конечным числом бит в памяти

Тип **double** требует 64 бита, которые распределены, как показано в таблице.

| Bits | Width | Usage |
|---|---|---|
| `63` | `1` | хранит знак, `0` положительный, `1` отрицательный |
| `62 to 52` | `11` | Хранит экспоненту, смещенную на `1023` |
| `51 to 0` | `52` | Хранит мантиссу |

```
class(1) % функция class(...) возвращает тип объекта
```

```
ans =
'double'
```

```
class(1.0)
```

```
ans =
```

```
'double'
```

```
1==1.0 % эквивалентно, то есть даже те числа, которые пишутся как целые матлабом
```

```
ans = logical
   1
```

```
% воспринимаются как числа с плавающей точкой (если никакого типа не
% задано)
1e3 % можно записывать в e-notation
```

```
ans = 1000
```

```
 num2hex(1) % представляет число в шестнадцатеричной системе
```

```
ans =
'3ff0000000000000'
```

```
num2hex(1.0)
```

```
ans =
'3ff0000000000000'
```

```
num2hex(1.0000000000000001)
```

```
ans =
'3ff0000000000000'
```

```
num2hex(1.0000000000000002)
```

```
ans =
'3ff0000000000001'
```

```
num2hex(realmin)
```

```
ans =
'0010000000000000'
```

```
1.0000000000000001==1
```

```
ans = logical
   1
```

```
1.0000000000000002==1
```

```
ans = logical
   0
```

```
eps % минимальная разница между числами с плавающей точкой
```

```
ans = 2.2204e-16
```

```
realmin % минимальное значение
```

```
ans = 2.2251e-308
```

```
realmax
```

```
ans = 1.7977e+308
```

```
a = realmax
```

```
a = 1.7977e+308
```

```
b = a+eps
```

```
b = 1.7977e+308
```

```
a-b
```

```
ans = 0
```

Особые числа inf и NaN

```
num2hex(Inf)
```

```
ans =
'7ff0000000000000'
```

```
num2hex(NaN)
```

```
ans =
'fff8000000000000'
```

```
%NaN==NaN
Inf==Inf
```

```
ans = logical
   1
```

```
bit_nan = num2hex(NaN)
```

```
bit_nan =
'fff8000000000000'
```

```
bit_nan(end)='f'
```

```
bit_nan =
'fff800000000000f'
```

```
hex2num(bit_nan)
```

```
ans = NaN
```

```
fit_inf = num2hex(inf)
```

```
fit_inf =
'7ff0000000000000'
```

```
fit_inf(1)='7'
```

```
fit_inf =
'7ff0000000000000'
```

```
hex2num(fit_inf)
```

```
ans = Inf
```

### Тип logial - основной логический тип `logical - 2 бита`

```
a = true
```

```
a = logical
   1
```

```
b= false
```

```
b = logical
   0
```

**Другие типы (об их существовании можно не знать до того как понадобится писать данные в бинарные файлы или работать с портами в бинарном режиме):**

- `single` - 32-бита
- `int8`, `int16`, `int32`, `int64`
- `uint8`, `uint16`, `uint32`, `uint64`
- `char - 8 бит`

## Массивы: вектора, матрицы и многомерные массивы чисел и логических элементов

Массив - конструкция для хранения данных одного типа.

Заполнения массивов вручную

```
clearvars % удаляет все переменные из памяти
a_row = [1,2,3,4] %  - строка
```

```
a_row = 1×4
     1     2     3     4
```

```
a_col = [1;2;3;4] %  - столбец
```

```
a_col = 4×1
     1
     2
     3
     4
```

```
a_row2 = [1 2 3 4]; %  - строка
a_lin = linspace(1,10,100); % фиксировано количество элементов (возвращает строку)
a_range = 1:0.01:100; % фиксированный шаг
a_range2 = 1:10; % если шаг не указанЮ будет единица
a_mat = [1,2,3,4;5,6,7,8] % матрица 2x4
```

```
a_mat = 2×4
     1     2     3     4
     5     6     7     8
```

Размерность массива

```
clearvars
a_scalar = 5 % массив размером (1,1)
```

```
a_scalar = 5
```

```
a_row = [1,2,3,4]; %  - строка
a_col = [1;2;3;4]; %  - столбец
a_mat = [1,2,3,4;5,6,7,8]; % матрица 2x4
size(a_row)
```

```
ans = 1×2
     1     4
```

```
size(a_col)
```

```
ans = 1×2
     4     1
```

```
numel(a_mat)
```

```
ans = 8
```

```
size(a_mat)
```

```
ans = 1×2
     2     4
```

```
size(a_mat,2) % количество столбцов
```

```
ans = 4
```

```
ndims(a_mat) % количество размерностей больших единицы
```

```
ans = 2
```

```
empty_mat = []
```

```
empty_mat =

     []
```

```
size(empty_mat)
```

```
ans = 1×2
     0     0
```

```
whos empty_mat % функция whos выдает информацию о переменных
```

```
  Name           Size            Bytes  Class     Attributes

  empty_mat      0x0                 0  double
```

## Индексирование

Прямое индексирование.

```
clearvars
a_rand = rand(10); % функция создает массив случайных чисел размером 10x10
a_rand(1,1) % (строка,столбец) декартово индексирование
```

```
ans = 0.8147
```

```
a_rand(100) % линейное (MATLAB - COLUMN ORIENTED LANGUAGE!)
```

```
ans = 0.3371
```

```
a_scalar = 5 % массив размером (1,1)
```

```
a_scalar = 5
```

```
a_scalar(1) % ВСЕ МАССИВ!
```

```
ans = 5
```

Индексирование при помощи "сахара": символы "**:**" и "**end**"

```
clearvars
a_rand = rand(10);
col1 = a_rand(:,2)% возвращает второй толбец
```

```
col1 = 10×1
    0.4505
    0.0838
    0.2290
    0.9133
    0.1524
    0.8258
    0.5383
    0.9961
    0.0782
    0.4427
```

```
row1 = a_rand(1,:)% возвращает первую строку
```

```
row1 = 1×10
    0.1622    0.4505    0.1067    0.4314    0.8530    0.4173    0.7803    0.2348 ···
```

```
a_mat = [1,2,3,4;5,6,7,8]
```

```
a_mat = 2×4
     1     2     3     4
     5     6     7     8
```

```
col = a_mat(:) % превращает матрицу в вектор-колонку
```

```
col = 8×1
     1
     5
     2
     6
     3
     7
     4
     8
```

```
submat = a_mat(2:end,:)
```

```
submat = 1×4
     5     6     7     8
```

Если при работе программы происходит изменение размера массива, то это сильно замедляет вычисления, чтобы этого избежать нужно резервировать память

Способы резервировать память

```
clearvars
a_zeros = zeros(10);
a_ones = ones(10,'double');
a_nan = NaN(10);% not-a-number
a_zeros5x5x10 = zeros([5 5 10]);
i_complex = 1i; % мнимая единица
a_complex = zeros(10,'like',i_complex); % удобно, когда в процессе работы не
известно какого типа будут данные  (пример - в функциях)
a_uint8 = zeros(10,'uint8');% можно указывать тип в явном виде
a_logical = false(10);% матрица 10x10 false
a_rand = rand(10);% матрица 10x10
```

BATTLE† - соревнование по скорости

timeit() - позволяет замерить время работы функции без входных аргументов

значок "@" - создает указатель на функцию (что это такое обсудим когда-нибудь потом), в данном случае нужен чтобы превратить функцию с аргументом в функцию без аргумента

BATTLE† Сравнение скорости заполнения матриц

```
clearvars
disp("Заполнение NaN'ами:")
```

```
Заполнение NaN'ами:
```

```
t1 = timeit(@()NaN(1000)) % заполнение NaN
```

```
t1 = 0.0019
```

```
disp("Заполнение нулями:")
```

```
Заполнение нулями:
```

```
t2 = timeit(@()zeros(1000)) % заполнение нулями
```

```
t2 = 8.2291e-06
```

```
disp("Заполнение inf'ами:")
```

```
Заполнение inf'ами:
```

```
t3 = timeit(@()inf(1000)) % заполнение инфами
```

```
t3 = 0.0018
```

```
disp("Заполнение единицами:")
```

```
Заполнение единицами:
```

```
t4 = timeit(@()ones(1000)) % заполнение единицами
```

```
t4 = 0.0019
```

BATTLE†Влияние преаллокации памяти на скорость вычислений

```
clearvars
disp("заполнение без преаллокации памяти:")
```

```
заполнение без преаллокации памяти:
```

```
timeit(@fill_by_column_no_memalloc) % no memory allocation
```

```
ans = 0.1919
```

```
disp("заполнение с преаллокацией памяти:")
```

```
заполнение с преаллокацией памяти:
```

```
timeit(@fill_by_column) % with memory allocation
```

```
ans = 0.0522
```

Заполнение матрицы из массива

```
clearvars
A = zeros(10);% memory allocation
a = linspace(1,100,100);
A(:) = a
```

```
A = 10×10
     1    11    21    31    41    51    61    71    81    91
     2    12    22    32    42    52    62    72    82    92
     3    13    23    33    43    53    63    73    83    93
     4    14    24    34    44    54    64    74    84    94
     5    15    25    35    45    55    65    75    85    95
     6    16    26    36    46    56    66    76    86    96
     7    17    27    37    47    57    67    77    87    97
     8    18    28    38    48    58    68    78    88    98
     9    19    29    39    49    59    69    79    89    99
    10    20    30    40    50    60    70    80    90   100
```

Индексирование при помощи массива

```
clearvars
a_col = [1;2;3;4]; %  - столбец
```

```
r_index = randi(4,100);% функция возвращает 100 случайных целых числа в интервале
от 1 до 4
a_permuted = a_col(r_index) % матрица из случаных перестановок элементов массива
a_col
```

```
a_permuted = 100×100
     1     2     3     4     3     3     4     3     4     3     2     3     1 ···
     3     1     3     3     4     1     1     2     3     2     3     3     2
     1     3     4     4     2     3     2     4     1     4     4     4     1
     1     2     2     3     2     1     2     1     4     2     4     3     2
     3     3     2     3     1     1     1     3     4     2     2     1     2
     1     3     1     4     1     3     3     3     4     4     4     2     2
     4     3     4     4     3     2     2     2     1     1     2     2     2
     4     1     3     4     2     3     4     1     2     4     4     4     1
     3     1     2     1     4     3     2     1     3     1     4     3     2
     1     2     3     4     1     3     3     2     3     3     4     3     3
     ⋮
```

Пример - вытаскивание диагонали:

```
clearvars
a_rand = rand(10); % генерим матрицу
col_size = size(a_rand,1); % число элементов в одном столбце
a_diagonal = a_rand(1:col_size+1:end); % индексирование при помощи диапазона с шагом
```

Удаление элемента при помощи индексирования

```
clearvars
a = 1:5;
a(2) = [] % изменяет матрицу in-place!
```

```
a = 1×4
     1     3     4     5
```

```
a(2:4)=[]; %???
```

Индексирование многомерных массивов

```
a_rand_3d = rand([10,10,10]);% трехмерная матрица
a_rand_3d(2,3,5) % элемент второй строки третьего столбца пятой страницы
```

```
ans = 0.8913
```

```
size(a_rand_3d(:,2,:))
```

```
ans = 1×3
    10     1    10
```

```
size(a_rand_3d(:))
```

```
ans = 1×2
        1000           1
```

Создание массива при индексировании:

```
clearvars
a_mat(10,10)=10
```

```
a_mat = 10×10
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0    10
```

BATTLE†Преаллокация памяти путем обратного индексирования:

```
disp("заполнение без преаллокации памяти:")
```

```
заполнение без преаллокации памяти:
```

```
timeit(@fill_by_column_no_memalloc) % no memory allocation
```

```
ans = 0.1901
```

```
disp("заполнение с преаллокацией памяти:")
```

```
заполнение с преаллокацией памяти:
```

```
timeit(@fill_by_column) % with memory allocation
```

```
ans = 0.0535
```

```
disp("заполнение с преаллокацией памяти путем обратного индексирования:")
```

```
заполнение с преаллокацией памяти путем обратного индексирования:
```

```
timeit(@fill_by_column_reverse_order)
```

```
ans = 0.0788
```

Таким образом: A(10:-1:1)=0, быстрее, чем A(1:1:10), так как в первом случае, преаллокация происходит один раз на первой итерации!

Но A = zeros([10,1]), все равно быстрее

Конкатенация матриц и векторов

```
clearvars
a = 1:5 % создаем вектор-строку
```

```
a = 1×5
     1     2     3     4     5
```

```
b = [a,10] % [.,.]  функции horzcat
```

```
b = 1×6
     1     2     3     4     5    10
```

```
b2 = horzcat(a,10)
```

```
b2 = 1×6
     1     2     3     4     5    10
```

```
c = [a,a]
```

```
c = 1×10
     1     2     3     4     5     1     2     3     4     5
```

```
v_col = [a(:),a(:)]
```

```
v_col = 5×2
     1     1
     2     2
     3     3
     4     4
     5     5
```

```
v_row = [a(:);a(:)]% [.;.]  функции vertcat
```

```
v_row = 10×1
     1
     2
     3
     4
     5
     1
     2
     3
     4
     5
```

```
v_row2 = [a;a]
```

```
v_row2 = 2×5
     1     2     3     4     5
     1     2     3     4     5
```

```
v_mat = [rand([5 2]),rand([5 2])]
```

```
v_mat = 5×4
    0.8357    0.2813    0.2832    0.1355
    0.3842    0.7813    0.7149    0.2728
    0.0095    0.5315    0.7344    0.6356
    0.3090    0.1539    0.4496    0.8062
    0.8367    0.5253    0.1821    0.6679
```

```
v_mat2 = [rand([5 2]);rand([5 2])]
```

```
v_mat2 = 10×2
    0.2866    0.4987
    0.1424    0.9860
    0.2459    0.9049
    0.6919    0.5752
    0.0717    0.7665
```

```
    0.7468    0.6189
    0.6953    0.8142
    0.6773    0.0160
    0.2824    0.2144
    0.5819    0.9879
```

## Символьные типы  char и string

```
clearvars
s_char = 'asfzassgfagag'; % ведут себя как массивы символов
s_string = "asfzassgfagag";% ведут себя как массивы групп символов
s_char(10)
```

```
ans =
'a'
```

```
%s_string(10)
s_char_pen = char(10000);
s_char_big = char(rand([1 10000]));
s_string_big = string(s_char_big);
onechar = 'a';
onestring = "a";
whos one*
```

```
  Name           Size            Bytes  Class     Attributes

  onechar        1x1                 2  char
  onestring      1x1               166  string
```

```
whos s_*
```

```
  Name             Size              Bytes  Class     Attributes

  s_char           1x13                 26  char
  s_char_big       1x10000           20000  char
  s_char_pen       1x1                   2  char
  s_string         1x1                 182  string
  s_string_big     1x1               20166  string
```

```
a_range_char = 'a':'z';
a_range_string = arrayfun(@(x)string(x),a_range_char);% создаем массив строк
"a"<="b"
```

```
ans = logical
   1
```

```
'a'<='b'
```

```
ans = logical
   1
```

```
s_char<s_char
```

```
ans = 1×13 logical array
   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Пустые массивы

```
clearvars
empty_char = ''
```

```
empty_char =

  0×0 empty char array
```

```
fake_empty_string = ""
```

```
fake_empty_string =
""
```

```
whos empty_char
```

```
  Name            Size            Bytes  Class    Attributes

  empty_char      0x0                 0  char
```

```
whos fake_empty_string
```

```
  Name                  Size            Bytes  Class     Attributes

  fake_empty_string     1x1               166  string
```

```
real_empty_string = strings(0,0)
```

```
real_empty_string =

  0×0 empty string array
```

```
whos real_empty_string
```

```
  Name                  Size            Bytes  Class     Attributes

  real_empty_string     0x0               112  string
```

BATTLE† Что быстрее строки или массивы символов?

```
[r_str,r_ch] = gen_random_string(1e6)% функция генерит случаную последовательность
1e5 символов
```

```
r_str =
"wuxufpntvjurrfdhdjyfhsdvtuwwantbgllwgdfgbqnrmhnxsmwwliigfihymrdlekovwdojhsyimtrgphdsqmrlexaoyemkxmrjixsumosqghmxqrx
r_ch =
'wuxufpntvjurrfdhdjyfhsdvtuwwantbgllwgdfgbqnrmhnxsmwwliigfihymrdlekovwdojhsyimtrgphdsqmrlexaoyemkxmrjixsumosqghmxqrx
```

```
disp("Поиск и замена паттерна в строке:")
```

```
Поиск и замена паттерна в строке:
```

```
timeit(@()replace(r_str,"a","I"))
```

```
ans = 0.0018
```

```
disp("Поиск и замена паттерна в массиве символов:")
```

Поиск и замена паттерна в массиве символов:

```
timeit(@()replace(r_ch,'a','I'))
```

```
ans = 0.0025
```

Конкатенация символьных типов

```
clearvars
s_char = 'a':'z'
```

```
s_char =
'abcdefghijklmnopqrstuvwxyz'
```

```
[s_char,s_char]
```

```
ans =
'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz'
```

```
[s_char;s_char]
```

```
ans = 2×26 char array
    'abcdefghijklmnopqrstuvwxyz'
    'abcdefghijklmnopqrstuvwxyz'
```

```
s_string = arrayfun(@(x)string(x),s_char);% создаем массив строк
[s_string;s_string]
```

```
ans = 2×26 string
"a"          "b"          "c"          "d"          "e"          "f"          "g···
"a"          "b"          "c"          "d"          "e"          "f"          "g
```

**Базовые операции над массивами**

Операции сравнения и логическое индексирование

```
clearvars
a = 1:10 % вектор-строка
```

```
a = 1×10
     1     2     3     4     5     6     7     8     9    10
```

```
% сравнение массива и скаляра
flag1 = a>3 % логический массив
```

```
flag1 = 1×10 logical array
   0   0   0   1   1   1   1   1   1   1
```

```
sum(flag1) % самый быстрый способ посчитать количество элементов массива,
удовлетворящих условию
```

```
ans = 7
```

```
b = randi(10,[1 10]) % вектор-строка случаных элементов
```

```
b = 1×10
```

```
     1     6     9     1     9     6     4     1     6     4
```

```
flag2 = a==b % проверяет элементы
```

```
flag2 = 1×10 logical array
   1   0   0   0   0   1   0   0   0   0
```

```
A_mat = rand(10)
```

```
A_mat = 10×10
    0.2809    0.9638    0.9308    0.4328    0.5977    0.2366    0.0992    0.4461 ···
    0.4957    0.3851    0.5260    0.4017    0.1345    0.2006    0.6706    0.1819
    0.8255    0.9450    0.2298    0.4931    0.4390    0.2034    0.2255    0.0584
    0.9708    0.4262    0.5193    0.0243    0.9214    0.1012    0.1560    0.6871
    0.5049    0.9621    0.1108    0.1073    0.2678    0.8808    0.9943    0.1225
    0.3785    0.8531    0.5942    0.0027    0.8001    0.4568    0.3518    0.1932
    0.2751    0.2384    0.1413    0.0410    0.5380    0.7363    0.8330    0.4732
    0.3400    0.2878    0.8747    0.1267    0.1229    0.4907    0.6864    0.5942
    0.2655    0.9337    0.6048    0.5010    0.5463    0.5975    0.2337    0.6335
    0.2312    0.1936    0.8778    0.0263    0.3614    0.6320    0.2751    0.6606
```

```
flag_mat = A_mat>0.5
```

```
flag_mat = 10×10 logical array
   0   1   1   0   1   0   0   0   1   1
   0   0   1   0   0   0   1   0   1   0
   1   1   0   0   0   0   0   0   1   0
   1   0   1   0   1   0   0   1   0   0
   1   1   0   0   0   1   1   0   1   0
   0   1   1   0   1   0   0   0   1   0
   0   0   0   0   1   1   1   0   0   0
   0   0   1   0   0   0   1   1   0   0
   0   1   1   1   1   1   0   1   1   0
   0   0   1   0   0   1   0   1   0   0
```

```
sum(flag_mat,"all") % суммирование по всем элементам
```

```
ans = 40
```

```
A_mat(flag_mat) % возвращает вектор-столбец
```

```
ans = 40×1
    0.8255
    0.9708
    0.5049
    0.9638
    0.9450
    0.9621
    0.8531
    0.9337
    0.9308
    0.5260
      ⋮
```

```
col_vec = randi(10,[5,1])
```

```
col_vec = 5×1
     9
     3
     4
     1
```

```
     6
```

```
row_vec = randi(10,[1,3])
```

```
row_vec = 1×3
     9    10     9
```

```
flag_col_row = col_vec>row_vec % ???
```

```
flag_col_row = 5×3 logical array
   0   0   0
   0   0   0
   0   0   0
   0   0   0
   0   0   0
```

```
flag_row_col = row_vec>col_vec % ???
```

```
flag_row_col = 5×3 logical array
   0   1   0
   1   1   1
   1   1   1
   1   1   1
   1   1   1
```

Как выбрать из массива элементы, удовлетворяющие неравенству:

```
clearvars
 a = rand(10,1)
```

```
a = 10×1
    0.1312
    0.5495
    0.1644
    0.5023
    0.4746
    0.8920
    0.4009
    0.8232
    0.5787
    0.0531
```

```
 flag1 = a>0.8
```

```
flag1 = 10×1 logical array
   0
   0
   0
   0
   0
   1
   0
   1
   0
   0
```

```
 flag2 = a<0.2
```

```
flag2 = 10×1 logical array
   1
```

```
   0
   1
   0
   0
   0
   0
   0
   0
   1
```

```
% логический плюс
flag_union = flag1 | flag2
```

```
flag_union = 10×1 logical array
   1
   0
   1
   0
   0
   1
   0
   1
   0
   1
```

```
%
flag_intersection = a>=0.2 & a<=0.5
```

```
flag_intersection = 10×1 logical array
   0
   0
   0
   0
   1
   0
   1
   0
   0
   0
```

```
a(flag_union) % элементы, которые либо <0.2, либо >0.8
```

```
ans = 5×1
    0.1312
    0.1644
    0.8920
    0.8232
    0.0531
```

```
a(flag_intersection) % элементы лежащие внутри [0.2,0.8]
```

```
ans = 2×1
    0.4746
    0.4009
```

```
% эквивалентно неравенству
```

# Выводы семинара 1.

1. Базовый матлаб - double  - число с плавающей точкой
2. Базовый объект  - массив.
3. Массивы поддерживают два основных метода работы a= M(i) - получить элемент массива (get_index), M(i)=a - задать элемент массива (set_index)
4. Массивы поддерживают линейное индексирование (один индекс) и декартово (число индексов равно размерности массива).
5. Можно индексировать при помощи массивов индексов и при помощи логических массивов
6. В отличие от большинства других языков, в матлаб метод set_index ( "имя_массива(индекс)=значение"  ) позволяет изменять размер массива в процессе выполнения программы, однако это медленно, поэтому нужно делать преаллокацию
7. Самый быстрый способ преаллокации - функция zeros(), можно делать преаллокацию обратным индексированием
8. Есть два базовых типа символьных объектов: 'char' - массив символов и "string" - массив групп символов, первый ест меньше памяти, второй немного быстрее

```
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 исоплъзования 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
```