

Лабораторная работа № 10

Работа с файлами в MatLab

В MatLab реализованы различные функции по работе с файлами, содержащие данные в самых разных форматах.

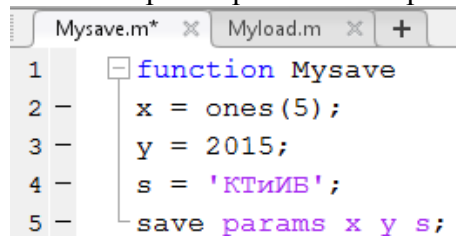
Функции save и load

В самом простом случае для сохранения и последующей загрузки каких-либо данных в MatLab предусмотрены две функции

save <имя файла> <имена переменных> % сохранение данных

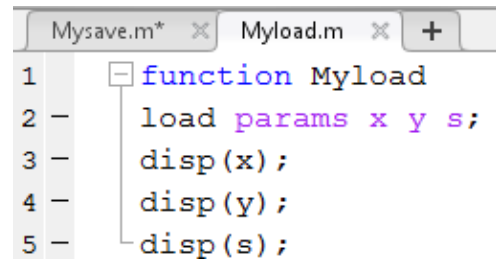
load <имя файла> <имена переменных> % загрузка данных

Функции save и load работают с файлом, который по умолчанию располагается в рабочем каталоге и имеет расширение mat. Пример:



```
1 function Mysave
2     x = ones(5);
3     y = 2015;
4     s = 'КТИИБ';
5     save params x y s;
```

Функция сохранения данных



```
1 function Myload
2     load params x y s;
3     disp(x);
4     disp(y);
5     disp(s);
```

Функция загрузки данных

При исполнении mat-файлов в консоли появится следующее:

```
>> Mysave
>> Myload

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

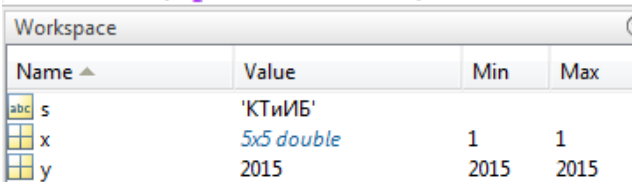
    2015

    КТИИБ
```

КТИИБ

В дальнейшем можно загружать данные файла params в Workspace:

```
>> load('params.mat')
```



Name	Value	Min	Max
s	'КТИИБ'		
x	5x5 double	1	1
y	2015	2015	2015

Следует обратить внимание, что функция load позволяет загружать из mat-файла не все, а только указанные программистом переменные, например:

load params x; % загружает только значение переменной x

Самостоятельно: Написать функцию, вычисляющую максимальное значение среди диагональных элементов заданной матрицы. Осуществить запись/чтение элементов матрицы в файл.

Недостатком рассмотренных функций save и load является то, что они работают с определенными форматами файлов (обычно mat-файлы) и не позволяют загружать или сохранять данные в других форматах.

Функции fwrite и fread

fwrite(<идентификатор файла>, <переменная>, <тип данных>);

и

<переменная>=fread(<идентификатор файла>);

<переменная>=fread(<идентификатор файла>, <размер>);

<переменная>=fread(<идентификатор файла>, <размер>, <точность>);

Здесь <идентификатор файла> - это указатель на файл, с которым предполагается работать. Чтобы получить идентификатор, используется функция
 <идентификатор файла> = fopen(<имя файла>, <режим работы>);
 где параметр <режим работы> может принимать значения, приведенные в табл. 1.

Таблица 1

Режимы работы с файлами в MatLab

параметр <режим работы>	описание
'r'	чтение
'w'	запись (стирает предыдущее содержимое файла)
'a'	добавление (создает файл, если его нет)
'r+'	чтение и запись (не создает файл, если его нет)
'w+'	чтение и запись (очищает прежнее содержимое или создает файл, если его нет)
'a+'	чтение и добавление (создает файл, если его нет)
'b'	дополнительный параметр, означающий работу с бинарными файлами, например, 'wb', 'rb', 'rb+', 'ab' и т.п.

Если функция fopen() по каким-либо причинам не может корректно открыть файл, то она возвращает значение -1. Ниже представлен фрагмент программы записи и считывания данных из бинарного файла:

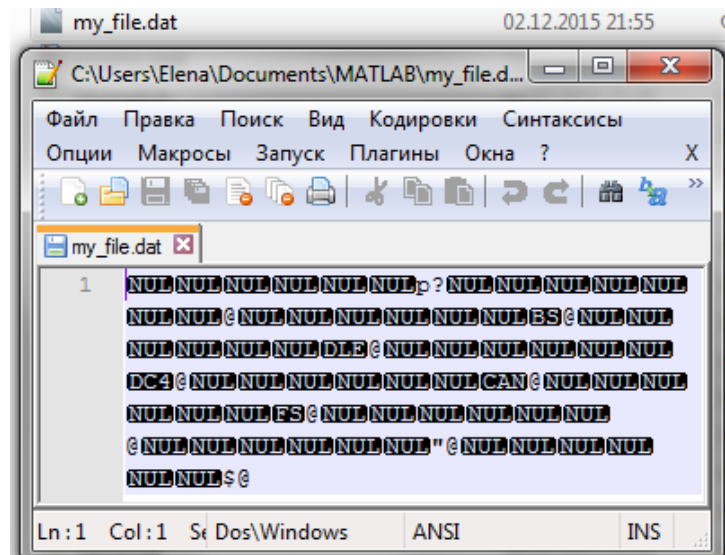
```

Binary.m*  x  +
1  function Binary
2  -    A = [1 2 3 4 5 6 7 8 9 10];
3
4      % открытие файла на запись
5  -    fid = fopen('my_file.dat', 'wb');
6  -    if fid == -1 % проверка корректности открытия
7  -        error('File is not opened');
8  -    end
9
10     % запись матрицы в файл
11 -    fwrite(fid, A, 'double');
12 -    fclose(fid); % закрытие файла
13
14     % открытие файла на чтение
15 -    fid = fopen('my_file.dat', 'rb');
16 -    if fid == -1
17 -        error('File is not opened');
18 -    end
19
20     % чтение 5 значений double
21 -    B = fread(fid, 5, 'double');
22 -    disp(B);
23 -    fclose(fid);

>> Binary
1
2
3
4
5

```

В результате работы данной программы в рабочем каталоге будет создан бинарный файл my_file.dat.



В приведенном примере явно указывалось число элементов (пять) для считывания из файла. В MatLab существует функция для проверки достижения конца файла:

`feof(<идентификатор файла>)`,

которая возвращает 1 при достижении конца файла и 0 в других случаях.

Пример, в котором динамически формируется вектор-строка по мере считывания элементов из входного файла:

```

function Read_inVector

fid = fopen('my_file.dat', 'rb'); % открытие файла на чтение
if fid == -1
    error('File is not opened');
end
B=0; % инициализация переменной
cnt=1; % инициализация счетчика
while ~feof(fid) % цикл, пока не достигнут конец файла
    [V,N] = fread(fid, 1, 'double');
    % считывание одного значения double
    % V содержит значение элемента,
    % N - число считанных элементов
    if N > 0 % если элемент был прочитан успешно, то
        B(cnt)=V; % формируем вектор-строку из значений V
        cnt=cnt+1; % увеличиваем счетчик на 1
    end
end
disp(B); % отображение результата на экран
fclose(fid); % закрытие файла

>> Read_inVector
    1     2     3     4     5     6     7     8     9    10

```

Функция `fread()` записана с двумя выходными параметрами `V` и `N`. Значение `N` будет равно 1 каждый раз при считывании информации из файла, и 0 при считывании служебного символа EOF, означающий конец файла.

Самостоятельно: Напишите функцию, вычисляющую сумму элементов каждой строки матрицы $|3 \times 3|$. Осуществить запись/чтение элементов матрицы в файл.

```

>> Sum_el_Matr
    1     2     3
    4     5     6
    7     8     9

    6    15    24

```

С помощью функций `fwrite()` и `fread()` можно сохранять и строковые данные. Например, пусть дана строка:

```
str = 'RSEU is the best university';
```

которую требуется сохранить в файл. В этом случае функция `fwrite()` будет иметь следующую запись:

```
fwrite(fid, str, 'int16');
```

Здесь используется тип `int16`, т.к. при работе с русскими буквами система MatLab использует двухбайтовое представление каждого символа.

Ниже представлена программа записи и чтения строковых данных.

```
function Str_job
fid = fopen('my_file2.dat', 'wb');
if fid == -1
    error('File is not opened');
end

str='RSEU is the best university'; % строка для записи
str1=' факультет КТиИБ лучший'; % строка для записи
% запись в файл
fwrite(fid, strcat(str, strcat(blanks(10), str1)), 'int16');
fclose(fid);

fid = fopen('my_file2.dat', 'rb');
if fid == -1
    error('File is not opened');
end

B=''; % инициализация строки
i=1;
while ~feof(fid)
    [V,N] = fread(fid, 1, 'int16=>char');
    % чтение текущего
    % символа и преобразование
    % его в тип char
    if N > 0
        B(i)=V;
        i=i+1;
    end
end
disp(B); % отображение строки на экране
fclose(fid);
```

Результат выполнения программы будет иметь вид:

```
>> Str_job
```

```
RSEU is the best university факультет КТиИБ лучший
```

Самостоятельно: Напишите функцию, записывающую/считывающую строку чисел в файл, разделенных пробелом. Вывести сумму числовых данных.

```
str='1 3 5';
```

```
>> Str_job
```

```
9
```

Функции `fscanf` и `fprintf`

Описанные выше функции работы с файлами позволяют записывать и считывать информацию по байтам, которые затем требуется правильно интерпретировать для преобразования их в числа или строки. В то же время выходными результатами многих программ являются текстовые файлы, в которых явным образом записаны те или числа или текст. Например, при экспорте данных из MS Excel можно получить файл формата (Data.dat):

```
174500,1.63820,1.63840,1.63660,1.63750,288
180000,1.63740,1.63950,1.63660,1.63820,361
181500,1.63830,1.63850,1.63680,1.63740,223
183000,1.63720,1.64030,1.63720,1.64020,220
```

где числа записаны в столбик и разделены запятой.

Прочитать такой файл побайтно, а затем интерпретировать полученные данные довольно трудоемкая задача, поэтому для этих целей были специально разработаны функции чтения

```
[value, count] = fscanf(fid, format, size)
```

и записи

```
count = fprintf(fid, format, a,b,...)
```

таких данных в файл. Здесь value – результат считывания данных из файла; count – число прочитанных (записанных) данных; fid – указатель на файл; format – формат чтения (записи) данных; size – максимальное число считываемых данных; a,b,.. – переменные для записи в файл.

Приведем пример чтения данных из файла, приведенного выше с помощью функции fscanf():

```
function fscanf_ex

fid = fopen('Data.dat', 'r');
if fid == -1
    error('File is not opened');
end

S = fscanf(fid, '%d,%f,%f,%f,%f,%d');
disp(S);
fclose(fid);
```

Здесь форматная строка состоит из спецификаторов и записана в виде '%d,%f,%f,%f,%f,%d'. Это означает, что сначала должно быть прочитано целочисленное значение из файла, затем, через запятую должно читаться второе вещественное значение, затем третье и так далее до последнего целочисленного значения. Полный список возможных спецификаторов приведен в табл. 2.

Таблица 2

Список основных спецификаторов для функций fscanf() и fprintf()

Спецификатор	Описание
%d	целочисленные значения
%f	вещественные значения
%s	строковые данные
%c	символьные данные
%u	беззнаковые целые значения

В результате работы программы переменная S будет представлять собой вектор-столбец:

```
S = [174500 1,6382 1,6384 1,6366 1,6375 288 180000 1,6374 1,6395 1,6366 1,6382
361 181500 1,6383 1,6385 1,6368 1,6374 223 183000 1,6372 1,6403 1,6372 1,6402 220]';
```

Несмотря на то, что данные были корректно считаны из файла, они из таблицы были преобразованы в вектор-столбец, что не соответствует исходному формату представления данных. Чтобы сохранить верный формат данных, функцию fscanf() в приведенном примере следует записать так:

```
S = fscanf(fid, '%d,%f,%f,%f,%f,%d', [6 4]);
```

Тогда на выходе получится матрица S размером в 6 строк и 4 столбца:

```
>> fscanf_ex
1.0e+05 *
```

```
1.7450    1.8000    1.8150    1.8300
0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000
0.0029    0.0036    0.0022    0.0022
```

Для записи данных в текстовый файл в заданном формате используется функция fprintf().

Будем предполагать, что матрица хранится в переменной Y.

```
fprintf(fid, '%6d;%.4f;%.4f;%.4f;%.4f;%d\r\n', Y');
```

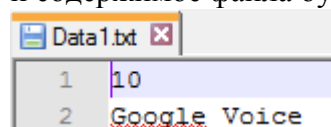
Следует отметить, что в функции fprintf() переменная Y имеет знак транспонирования ', т.к. данные в файл записываются по столбцам матрицы. Кроме того, перед спецификаторами стоят числа, которые указывают, сколько значащих цифр числа должно быть записано в файл. Например, спецификатор %6d говорит о том, что целые числа должны иметь 6 значащих цифр, а спецификатор %.4f означает, что после запятой будет отображено только 4 цифры. Наконец, в форматной строке были использованы управляющие символы \r – возврат каретки; \n – переход на новую строку. В итоге, содержимое файла будет иметь вид:

```
180000;1.2821;1.2824;1.2810;1.2812;490
190000;1.2810;1.2815;1.2798;1.2807;444
200000;1.2805;1.2808;1.2798;1.2800;399
210000;1.2799;1.2802;1.2788;1.2797;408
220000;1.2798;1.2806;1.2788;1.2803;437
230000;1.2804;1.2817;1.2802;1.2813;419
```

С помощью функции fprintf() можно записать значения двух и более переменных разного формата. Например, для записи числа и строки можно воспользоваться следующей записью:

```
function fprintf_ex2
fid = fopen('Data1.txt', 'w');
if fid == -1
    error('File is not opened');
end
str = 'Google Voice1';
y = 10;
fprintf(fid, '%d\r\n%s\r\n', y, str);
fclose(fid);
```

и содержимое файла будет иметь вид:



```
1 10
2 Google Voice
```

Функции imread и imwrite

При работе с файлами изображений, представленных в форматах bmp, png, gif, jpeg, tif и т.д., используются функции чтения

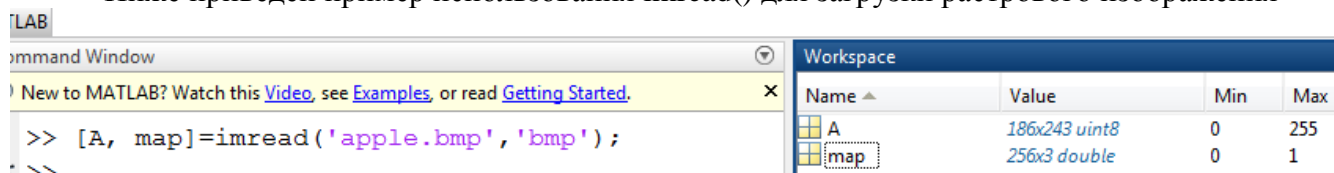
```
[X, map] = imread(filename, fmt)
```

и записи

```
imwrite(X, map, filename, fmt)
```

Здесь X – матрица точек изображения; map – цветовая карта изображения; filename – путь к файлу; fmt – графический формат файла изображения.

Ниже приведен пример использования imread() для загрузки растрового изображения



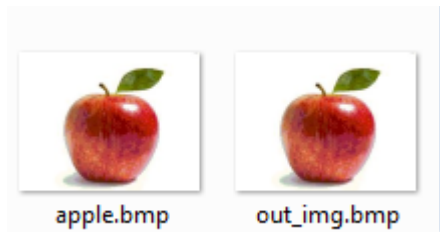
```
>> [A, map]=imread('apple.bmp', 'bmp');
```

Name	Value	Min	Max
A	186x243 uint8	0	255
map	256x3 double	0	1

где A – матрица размером 186x243xN точек; map – цветовая карта загруженного изображения. Значение N показывает число байт, необходимых для представления точки изображения. Например, если изображение представляется в формате RGB с 24 бит/пиксел, то N=3. Если же загружается изображение с 256 градациями серого (8 бит/пиксел), то N=1.

После обработки изображение A можно обратно сохранить в файл, используя следующую запись:

```
imwrite(A, map, 'out_img.bmp', 'bmp');
```



Однако следует отметить, что если загруженное изображение A было преобразовано, например, в формат `double`

```
A = double(A);
```

то непосредственная запись такой матрицы как изображение невозможно. Дело в том, что значения матрицы A должны соответствовать целым числам в диапазоне от 0 до 255, т.е. являться байтовыми числами. Этого можно добиться преобразованием типов при записи изображения в файл следующим образом:

```
imwrite(uint8(A), map, 'out_img.bmp', 'bmp');
```

Здесь `uint8` – беззнаковый целый тип в 8 бит.

В качестве переменной `map` можно указывать любые другие цветовые карты (`hot`, `hsv`, `gray`, `pink`, `cool`, `bone`, `copper`), отличные от исходной. Например, для записи изображения в 256 градациях серого можно записать

```
imwrite(uint8(A), gray(256), 'out_img.bmp', 'bmp');
```

При этом матрица A должна иметь размерность $M \times N \times 1$, т.е. один байт на пиксел.



Самостоятельно: Создайте копии изображения с разными цветовыми картами.

Для отображения растровых изображений в графическом окне MatLab используйте функцию `image(A)`;

Неверное отображение изображения объясняется несоответствием палитры цветов по умолчанию (`hot`), заданное в MatLab. Для замены одной палитры на другую используйте функцию `colormap(gray(256))`;

Если цветовая палитра заранее неизвестна на момент загрузки изображения, то ее можно узнать, используя второй возвращаемый параметр функции `imread`: `colormap(map)`;

При работе с изображениями возникают ситуации, когда диапазон значений элементов матрицы A может не соответствовать диапазону значений цветовой карты. В результате отображения такой матрицы на экране монитора изображение будет показываться некорректно. Чтобы избежать такой ситуации, диапазон значений и диапазон цветовой карты должны совпадать. Это можно сделать искусственно, масштабируя соответствующим образом значения элементов матрицы A . Однако, MatLab предоставляет функцию

```
imagesc(A);
```

которая делает это автоматически. Благодаря ее использованию, масштаб значений матрицы A всегда будет приведен к масштабу цветовой карты.