Основы программирования в MatLab.

Программы на языке MatLAB имеют две разновидности - **Script-файлы** (файлы-сценарии, или управляющие программы) и **файлы-функции** (процедуры).

С помощью **Script-файлов** оформляют основные программы, управляющие от начала до конца организацией всего вычислительного процесса, и отдельные части основных программ (они могут быть записаны в виде отдельных Script-файлов).

Главным внешним отличием текстов этих двух видов программ является то, что **файл-функции** имеют первую строку вида

`function <пвв> = <имя процедуры >(<пвп>),`

где **пвв** - перечень **возвращаемых** величин,

пвп - перечень **входных** параметров.

Script-файлы такой строки не имеют.

Принципиальное же отличие состоит в различном восприятии системой имен переменных в этих двух видах файлов

Типы программ используемых в системе Matlab

В файл - функциях все имена переменных внутри файла, а также имена переменных, указанные в заголовке (**ПВВ** и **ПВП**), воспринимаются как ***локальные***, т.е. все значения этих переменных после завершения работы процедуры исчезают, и область оперативной памяти ПК, которая была отведена под запись значений этих переменных, освобождается для записи в нее значений других переменных.

В Script-файлах все используемые переменные образуют так называемое ***рабочее пространство*** *(Workspace)*. Значение и набор их сохраняются не только на протяжении времени работы программы, но и на протяжении всего сеанса работы с системой, а, значит, и при переходе от выполнения одного Script-файла к другому.

Иначе говоря, рабочее пространство является единым для всех Script-файлов, вызываемых в текущем сеансе работы с системой.

Благодаря этому любой длинный Script-файл можно разбить на несколько фрагментов, оформить каждый из них в виде отдельного Script-файла, а в главном Script-файле вместо соответствующего фрагмента записать оператор вызова Script-файла, представляющего этот фрагмент. Этим обеспечивается компактное и наглядное представление даже довольно сложной программы.

За исключением указанных отличий, файл функции и Script-файлы

Особенности оформления m-файлов.

Под ***m-файлом*** будем понимать любой файл (***файл-функцию*** или ***Script-файл***), записанный на языке системы MatLAB.

1. Обычно каждый оператор записывается в отдельной строке текста программы. Признаком конца оператора является символ возврата каретки и перехода на следующую строку, который вводится в программу при нажатии клавиши <Enter>, т. е. при переходе на следующую строку.

2. Можно размещать несколько операторов в одной строке. Тогда предыдущий оператор этой строки должен заканчиваться символом ' **;** ' или ' **,** '.

3. Можно длинный оператор записывать в несколько строк. При этом предыдущая строка оператора должна заканчиваться тремя точками (' **...** ').

4. Если очередной оператор не заканчивается символом ' **;** ', результат его действия при выполнении программы будет выведен в командное окно.

5. Строка программы, начинающаяся с символа ' **%** ', не выполняется. Эта строка воспринимается системой MatLAB как комментарий.

6. Строки комментария, предшествующие первому выполняемому оператору программы, т. е. такому, который не является комментарием, воспринимаются системой MatLAB как ***описание программы***. Именно эти строки выводятся в командное окно, если в нем набрана команда **help <имя файла>**

Особенности оформления m-файлов.

7. В программах на языке MatLAB отсутствует ***символ окончания текста программы***.

8. В языке MatLAB ***переменные не описываются и не объявляются***. Любое новое имя, появляющееся в тексте программы при ее выполнении, воспринимается системой MatLAB как имя матрицы. Размер этой матрицы устанавливается при предварительном вводе значений ее элементов либо определяется действиями по установлению значений ее элементов, описанными в предшествующих операторах или процедуре. Эта особенность делает язык MatLAB очень простым в употреблении и привлекательным. В языке MatLab невозможно использование матрицы или переменной, в которой предварительно не введены или не вычислены значения ее элементов (а значит - и не определены размеры этой матрицы). В этом случае при выполнении программы MatLAB появится сообщение об ошибке - **"Переменная не определена".**

9. Имена переменных могут содержать лишь буквы латинского алфавита или цифры и должны начинаться из буквы. Общее число символов в имени может достигать **30**. В именах переменных могут использоваться как прописные, так и строчные буквы. ***Строчные и прописные буквы в именах различаются системой.*** Например, символы "**a**" и "**A**" могут использоваться в одной программе для обозначения разных величин.

Создание Script-файлов.

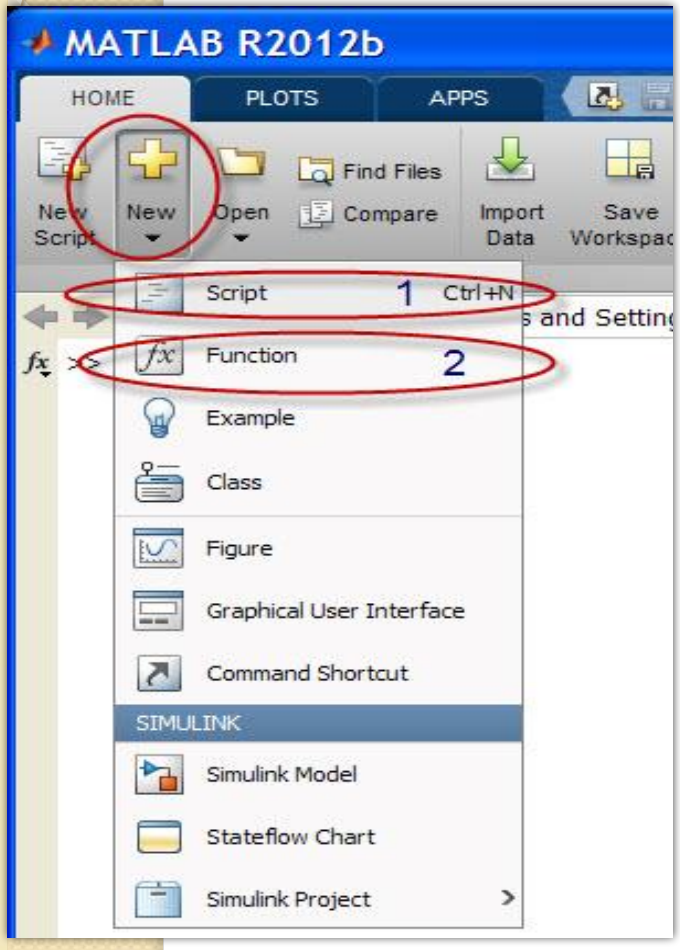
- ***Script-файлы являются независимо (самостоятельно) испол-няемыми блоками операторов и команд;***
- ***все используемые переменные образуют так называемое рабочее пространство, которое является общим для всех исполняемых Script-файлов; из этого следует, что при выполнении нескольких Script-файлов имена переменных в них должны быть согласованы, так как одно имя означает в каждом из них один и тот же объект вычислений;***
- ***в них отсутствует заголовок, т. е. первая строка не имеет определённого вида и назначения;***
- ***обращение к ним не требует указания имён переменных: все переменные формируются в результате выполнения прог-раммы либо сформированы ранее и существуют в рабочем пространстве.***

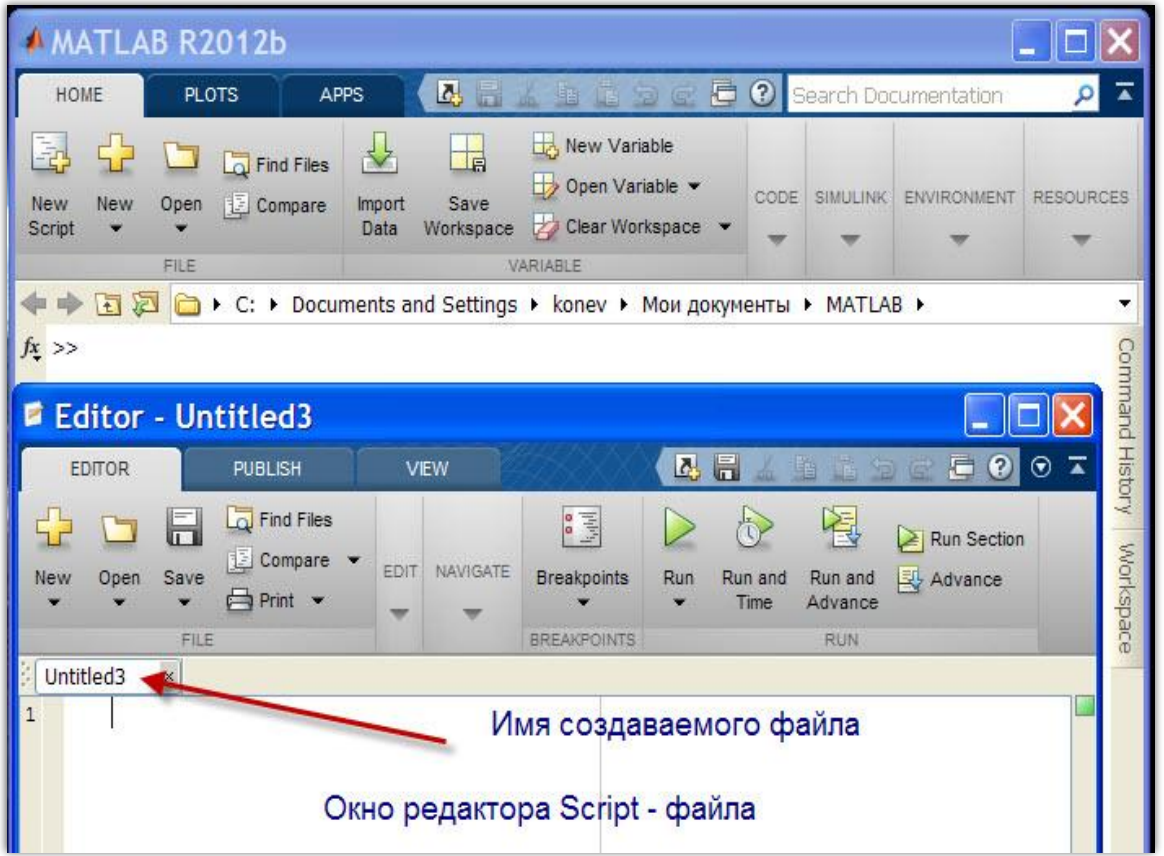
Рабочее пространство Script-файлов недоступно для файлов-функций, которые используются в нем

В **файлах-функциях** невозможно использовать значения, которые приобретают переменные в **Script-файле**, обходя заголовок файл-функции (так как все переменные файл-функции являются ***локальными***).

Создание Script-файлов.

Последовательность действий для открытия редактора Script – файла.





Типовая структура и оформление Script-файла.

```
% <Идентификатор(имя)Script-файла (ScrFil.m)>
% <Текст комментария с описанием назначения программы>
% Автор < Фамилия И. О., дата создания, организация>
< Пустая строка >
ScrFil_Zastavka
k = menu(' Что делать ? ','Продолжить работу ', ' Закончить работу ');
if k==1,
   while k==1
      ScrFil_Menu
      ScrFile_Yadro
      k = menu('Что делать ?','Продолжить работу', 'Закончить работу');
   end
end
ScrFil_Kin
```

ScrFil_Zastavka – имя отдельного **Script**- файла, выводящего на экран информацию о файле, либо меню для ввода исходных данных, …

ScrFil_Kin - имя отдельного **Script**- файла, в котором могут быть запрограммированы действия по завершению работы программы, например, очистить рабочее пространство от введённых глобальных переменных (оставаясь в рабочем пространстве, они препятствуют корректной работе другой программы, которая может иметь другие глобальные переменные, или переменные с теми же именами, но иными по типу, смыслу и значению), закрыть открытые программой графические окна (фигуры) и т.д.

Создание файл-функций (процедур). Общие требования

Файл-функция (***процедура***) должна начинаться со строки заголовка.

`function [<пкв>] = <имя процедуры>(<пвв>).`

Если перечень конечных (выходных) величин (**пкв**) содержит только один объект (***в общем случае - матрицу***), то файл-функция представляет собой обычную функцию (одной или нескольких переменных). Первая строка в этом случае имеет вид:

`function <имя переменной> = <имя процедуры>(<ПВВ>).`

Если же в результате выполнения файл-функции должны быть определены (вычислены) несколько объектов (***матриц***), такая файл-функция представляет собой уже более сложный объект, который в программировании обычно называется процедурой (в языке Паскаль), или подпрограммой. Общий вид первой строки в этом случае становится таким:

`function [y1, y2, ... , yn] = <имя процедуры>(<пвв>),`

т. е. перечень выходных величин **`y1, y2, ... , yn`** должен быть представлен как вектор-строка с элементами **`y1, y2, ... , yn`** (все они могут быть матрицами).

В простейшем случае заголовок функции одной переменной приобретёт вид:

`function y = func(x)`

где **`func`** - имя функции (m-файла)

Создание файл-функций (процедур).

Пример: Рассмотрим составление **m-файла** для вычисления функции

$$y(x) = d^3 \cdot ctg(x) \cdot \sqrt{\sin^4(x) - \cos^4(x)}$$

Для открытия редактора файла-функции в MatLAB 8.0 следует активизировать меню <**HOME**> <**New**> < **Function** >. На экране появится окно текстового редактора. В нем нужно набрать текст:

```
function [y] = Fun1(x,d)
%Процедура, которая вычисляет значение функции
%  y = (d3)*ctg(x)*sqrt(sin(x)4-cos(x)4).
%              Обращение y = F1(x,d).
y = (d^3).*cot(x).*sqrt(sin(x).^4-cos(x).^4);
```

После этого необходимо сохранить этот текст в файле под именем (допустим) **Fun1.m** ***в вашем активном каталоге.***

Теперь можно пользоваться этой функцией при расчётах. Так, если ввести команду в окне Command Window

» **y = Fun1(1, 0.1)** то получим результат y = 4. 1421e-04.

Создание файл-функций (процедур).

Следует заметить, что аналогично можно получить сразу ***вектор*** всех значений указанной функции при разных значениях аргумента Так, если сформировать вектор **zet** и обратиться в ту же процедуру, то получим:

```
» zet= 1:0.2:2.0;
» my = Fun331(zet,1)
my =
   0.4142    0.3339    0.1674   -0.0292   -0.2209   -0.3700
```

*1. Возможность использования созданной процедуры как для **отдельных чисел**, так и для **векторов и матриц** обусловлена применением в записи m-файла вместо обычных знаков арифметических действий их аналогов с предшествующей точкой* (**.* ./ .^**)

*2. Во избежание вывода на экран нежелательных промежуточных результатов, необходимо в тексте процедуры все вычислительные операторы завершать символом " **;** ".*

3. ***Важно, чтобы структура обращения полностью соответствовала структуре заголовка в записи текста М-файла и чтобы переменные в этом обращении имели тот же тип и размер, как и в заголовке М-файла.***

Создание файл-функций (процедур).

Пример: Создадим файл-функцию, вычисляющую значения функции

$$y(t) = k1 + k2 \cdot t + k3 \cdot \sin(k4 \cdot t + k5)$$

В этом случае удобно объединить совокупность коэффициентов **k** в единый вектор-строку **K = [k1 k2 k3 k4 k5]** и создать такой **m**-файл:

```
function [y] = Fun333(x,K)
% Вычисление функции y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5)),
%                    где K - вектор из пяти элементов
y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5));
end
```

Тогда расчёт, например, 11-ти значений этой функции можно осуществить так

```
» K = ones(1,5);
» t = 0:1:10;
» fi = Fun333(t, K)
fi =
```

1. 8415 2. 9093 3. 1411 3. 2432 4. 0411 5. 7206 7. 6570 8. 9894 9. 4560 10. 0000

Типовое оформление процедуры-функции.

Рекомендуется оформлять m-файл процедуры-функции по следующей схеме:

```
function [<Выходные парам.>] = <имя функции>(<Входные парам.>)
% <Краткое пояснение назначения процедуры>
%  Входные переменные
% <Детальное пояснение о назначении, типе и размерах
%          каждой из переменных, перечисленных в перечне <Входные парам.>
%  Выходные переменные
% <Детальное пояснение о назначении, типе и размерах
%                каждой из переменных перечня <Выходные парам.>
%                и величин, используемых в процедуре как глобальные>
%  Использование других функций и процедур
% <Раздел заполняется, если процедура содержит обращение
%            к другим процедурам, кроме встроенных процедур MatLab>
< Пустая строка >
% Автор : <Указывается автор процедуры, дата создания конечного варианта
%                процедуры и организация, в которой созданная программа>
< Текст исполняемой части процедуры >
end
```

Перечень входных и выходных переменных, разделяется запятыми.

При использовании команды **`help <имя процедуры>`** в командное окно выводятся строки комментария до первой **пустой строки**.

Чтение и запись данных в Matlab.

Функции save и load

Создание программ часто предполагает сохранение результатов расчетов в файлы для их дальнейшего анализа, обработки, хранения и т.п. В связи с этим в MatLab реализованы различные функции по работе с файлами, содержащие данные в самых разных форматах.

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

```
x=ones(5);
y=5;
s='hello';

save('File01','x','y','s');
x=zeros(5);
y=0;
s='';

load('File01','x','y','s');
disp(x);
disp(y);
disp(s);
```

Функция **save** позволяет сохранять произвольные переменные программы в файл, который будет (по умолчанию) располагаться в ***рабочем каталоге*** и иметь расширение **mat**. Соответственно функция **load** позволяет загрузить из указанного mat-файла ранее сохраненные переменные.

Функция **load** позволяет загружать из mat-файла не все, а только указанные переменные.

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

Чтение и запись данных в Matlab.

Функции fwrite и fread

На практике возникает необходимость загружать информацию, например, из **бинарных файлов**, созданных другими программными продуктами для дальнейшей обработки результатов в MatLab. С этой целью были разработаны функции

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

и

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

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

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

где <**идентификатор файла**> - это указатель на файл, с которым предполагается работать. Чтобы объявить идентификатор, используется функция

<идентификатор файла> = fopen(<имя файла>,<режим работы>);

где параметр <**режим работы**> может принимать значения

`'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)

Чтение и запись данных в Matlab.

Функции fwrite и fread

'a+' чтение и добавление (создает файл, если его нет)

'b' дополнительный параметр, означающий работу с бинарными файлами, например, '**wb**', '**rb**', '**rb+**', '**ab**' и др.

Если функция **fopen**() по каким-либо причинам не может корректно открыть файл, то она возвращает значение **-1**.

Фрагмент программы записи и считывания данных из бинарного файла

```
  A = [1 2 3 4 5];
fid = fopen('my_file.dat','wb'); % открытие файла на запись
if fid == -1                     % проверка корректности открытия
    error('File is not opened');
end
fwrite(fid, A, 'double');        % запись матрицы в файл (40 байт)
fclose(fid);                     % закрытие файла
fid = fopen('my_file.dat','rb'); % открытие файла на чтение
if fid == -1                     % проверка корректности открытия
  error('File is not opened');
end
 B = fread(fid,5,'double');      % чтение 5 значений double
disp(B);                         % отображение на экране
```

Чтение и запись данных в Matlab.

Функции fwrite и fread

С помощью функций **fwrite**() и **fread**() можно сохранять и строковые данные.

```
str = 'Hello MatLab';
```

требуется сохранить эту строку в файл. Функция **fwrite**() будет иметь следующую запись:

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

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, используя функции fwrite() и fread():

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

str='Привет MatLab';                % строка для записи
fwrite(fid, str, 'int16');          % запись в файл
fclose(fid);

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

Чтение и запись данных в Matlab.

Функции fwrite и fread

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

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

Результат выполнения программы будет иметь вид

Чтение и запись данных в текстовые файлы.

Текстовые файлы данных отличаются от **бинарных** прежде всего тем, что информация в них содержится в виде закодированных текстовых символов, т. е. в **символьном виде**.

В число записываемых символов входят и такие явно не регистрируемые символы, как символ окончания строки, перевода каретки, абзаца и др. Поэтому текстовые файлы представляют для текстовых редакторов **сформированный текстовый фрагмент**.

Текстовые файлы пригодны и для записи чисел, если предварительно преобразовать эти числа в символьное представление.

Форматы символьного представления чисел в MatLAB :

Short (default) - краткая запись (применяется по умолчанию) с фиксированной запятой;

Long - длинная запись с фиксированной запятой;

Short E - краткая запись в формате с плавающей запятой;

Long E - длинная запись в формате с плавающей запятой;

Short G - вторая форма краткой записи в формате с плавающей запятой;

Long G - вторая форма длинной записи в формате с плавающей запятой;

Hex - запись в виде шестнадцатеричного числа;

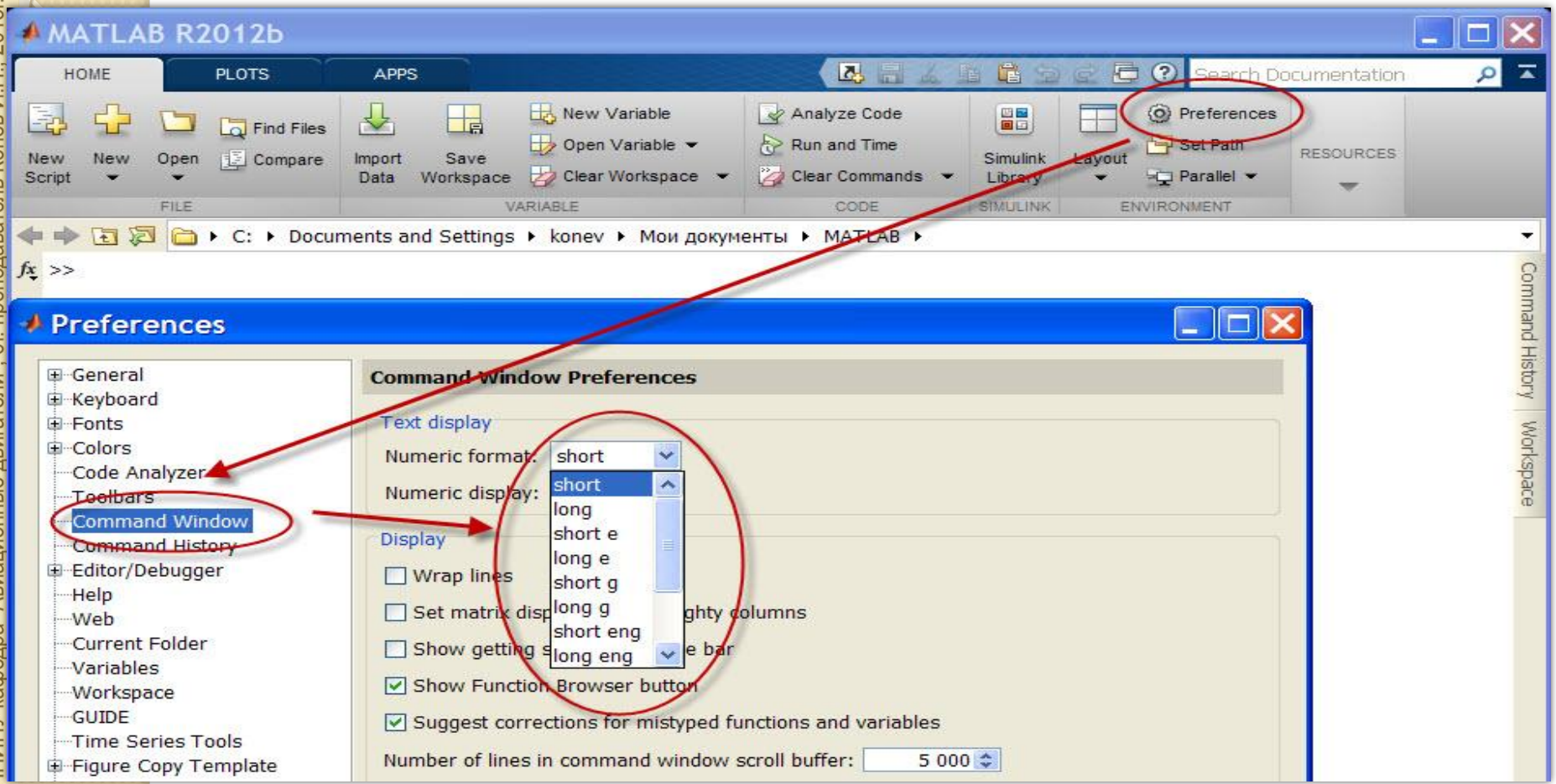
Bank - запись до сотых долей;

+ - записывается только знак числа;

Rational - запись в виде рациональной дроби.

Чтение и запись данных в текстовые файлы.

Формат вывода чисел на экран монитора может быть установлен с помощью команд меню <**Preferences**> <**Command Window**>.



Чтение и запись данных в текстовые файлы.

Функции fscanf и fprintf

Запись данных в текстовый файл осуществляется применением функции fprintf

fprintf ('<**имя_файла**>','**строка управляющих символов**',<**ПЗВ**>)

Здесь <**имя_файла**> - имя файла, в который записываются данные;

<**ПЗВ**> - перечень записываемых величин (они должны быть заданы (определены) до открытия файла для записи).

Строка управляющих символов (она должна быть заключена в апострофы) содержит информацию о том, в каком формате будут записываться данные, указанные в <**ПЗВ**>. Она может содержать, помимо специальных управляющих символов и произвольные обычные символы. Тогда эти символы будут помещены между записываемыми данными.

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

%f	очередная переменная будет записана как действительное число в форме с фиксированной десятичной запятой; между символами **%** и **f** могут быть записаны два целых числа и разделяющая их точка; первое число задает полное количество символов, отводимых на запись числа, второе – число символов после десятичной точки;
%g	числа в форме с плавающей десятичной запятой
%s	строковые данные

Чтение и запись данных в текстовые файлы.

Функции fscanf и fprintf

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

%u	беззнаковые целые значения
%d	целочисленные значения
%c	символьные данные
\n	конец строки и перевод каретки на следующую строку
\t	вставка горизонтальной табуляции
\r	перевод каретки на начало строки
\b	возврат на один символ
\f	переход к новой странице
\” или **‘ ‘**	проставить знак апострофа
%%	проставить знак процента

Чтение и запись данных в текстовые файлы.

Функции fscanf и fprintf

Формат обращения к функциям чтения и записи текстовых файлов

```
[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 (аналог файловой переменной в Pascal);

format – формат чтения (записи) данных;

size – максимальное число считываемых данных;

a,b,.. – переменные для записи в файл.

Перед выполнением операций чтения (записи) должна быть выполнена операция открытия файла.

```
fid = fopen('my_filel.txt','w');
  if fid == -1
          error('File is not opened');
     end
```

Чтение и запись данных в текстовые файлы.

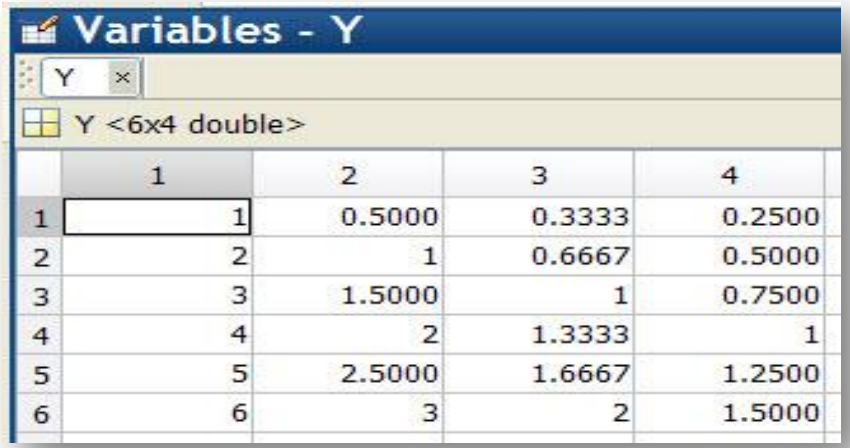
Функции `fscanf` и `fprintf`

Пример: ***Запишем матрицу чисел в файл, в котором числовые значения должны разделяться точкой с запятой.***

1. Создадим прямоугольную матрицу **Y(4x6)** (6 строк , 4 столбца).

```
>> for i=1:6
for j=1:4
      Y(i,j)=i/j;
end
end
disp([Y])
```

```
    1.0000    0.5000    0.3333    0.2500
    2.0000    1.0000    0.6667    0.5000
    3.0000    1.5000    1.0000    0.7500
    4.0000    2.0000    1.3333    1.0000
    5.0000    2.5000    1.6667    1.2500
    6.0000    3.0000    2.0000    1.5000
```



	1	2	3	4
1	1	0.5000	0.3333	0.2500
2	2	1	0.6667	0.5000
3	3	1.5000	1	0.7500
4	4	2	1.3333	1
5	5	2.5000	1.6667	1.2500
6	6	3	2	1.5000

Чтение и запись данных в текстовые файлы.

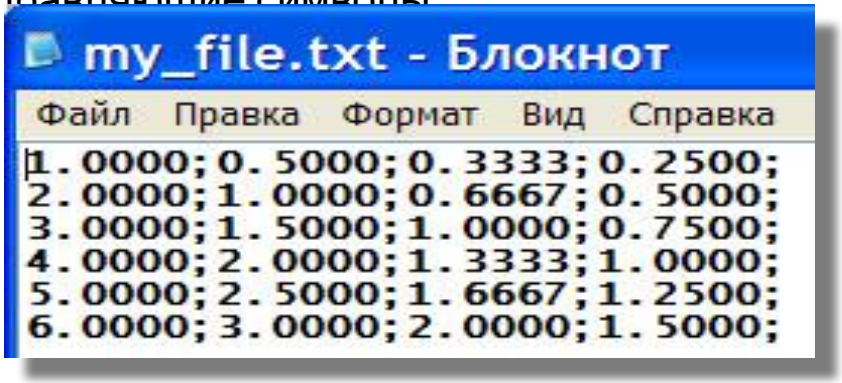
```
>> fid = fopen('my_file.txt', 'w');
>> if fid == -1
error('File is not opened');
end
>> fprintf(fid, '%.4f;%.4f;%.4f;%.4f;\r\n', Y');
>> fclose(fid);
```

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 **%.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

которые необходимы для формирования строк в выходном файле.



Чтение и запись данных в текстовые файлы.

Пример чтения данных из файла, с помощью функции **fscanf**():

При экспорте данных из **MS Excel** можно получить файл формата

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

```
fid = fopen('my_filel.txt','w');
  if fid == -1
        error('File is not opened');
  end
S = fscanf(fid, '%d,%f,%f,%f,%f,%d');
fclose(fid);
```

Строка спецификаторов **'%d,%f,%f,%f,%f,%d'** 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.

Чтение и запись данных в текстовые файлы.

В результате работы программы переменная **S** будет представлять собой ***вектор-столбец***, состоящий из **24** элементов:

```
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 столбцов и 4 строки с соответ-ствующими числовыми значениями.

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**