

# Тема 6. Программирование в среде MatLAB

## 6.1. Создание M-файлов

Для создания M-файла необходимо из меню *File* командного окна выбрать *New > M-file*. После чего появляется окно редактора (*M-file Editor*).

Программы на языке MatLAB имеют две разновидности – **Script-файлы** (*файлы-сценарии, или управляющие программы*) и **файлы-функции** (*процедуры*). Обе разновидности должны иметь расширение имени файла *.m*.

С помощью **Script-файлов** оформляют основные программы, управляющие от начала до конца организацией всего вычислительного процесса, а также отдельные части основных программ.

**Файл-функции** служат для оформления отдельных процедур и функций (т. е. частей программы, рассчитанных на неоднократное использование Script-файлами при измененных значениях исходных параметров и не могут быть выполнены без предварительного задания значений входных переменных).



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

**function** <ПКВ> = <имя процедуры >(<ПВВ>)

где обозначен ПКВ – Перечень Конечных Величин, ПВВ – Перечень Входных Величин. *Script-файлы* такой строки не имеют.

Принципиальное же отличие состоит в совсем различном восприятии системой имен переменных.


В **файл-функциях** все имена переменных внутри файла, а также имена переменных, указанные в заголовке (ПКВ и ПВВ), воспринимаются как локальные, т.е. все значения этих переменных после завершения работы процедуры исчезают.

В **Script-файлах** все используемые переменные образуют рабочее пространство (Work Space). Значение и содержание их сохраняются не только на протяжении времени работы программы, но и на протяжении всего сеанса работы с системой.

## 6.2. Особенности оформления М-файлов

1. Обычно каждый оператор записывается в отдельной строке текста программы.
2. При размещении нескольких операторов в одной строке 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 воспринимаются системой как описание программы. Именно эти строки выводятся в командное окно, если в нем набрана команда

***help*** <имя файла>

- 
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 как имя матрицы. Размер этой матрицы устанавливается предварительно.*
  9. В языке MatLAB *невозможно использование матрицы или переменной, в которой предварительно не введены или не вычислены значения ее элементов. 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 с буквы. Общее число символов в имени может достигать 30. В именах переменных могут использоваться как прописные, так и строчные буквы. Особенностью языка MatLAB есть то, что *строчные и прописные буквы в именах различаются системой.*



### 6.3. Создание простейших *файл-функций* (процедур)

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

***function*** [<ПКВ>] = <имя процедуры>(<ПВВ>)

Если перечень конечных (выходных) величин (ПКВ) содержит только один объект, 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 несколько объектов (матриц), то такая файл-функция представляет собой процедуру (подпрограмму). Общий вид первой строки в этом случае становится таким:

***function*** [y1, y2, ... , y] = <имя процедуры>(<ПВВ>)

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

В простейшем случае функции одной переменной заголовок приобретет вид:

$$\mathbf{function\ } y = \mathbf{func}(x)$$

где **func** – имя функции (M-файла).

В качестве примера рассмотрим составление M-файла для функции

$$y = f_1(x) = d^3 \cdot ctg(x) \cdot \sqrt{\sin^4(x) - \cos^4(x)}$$

В окне текстового редактора нужно набрать такой текст:

```
function y = F1(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 F1.m.

Теперь введём команду

```
» y = F1(1, 0.1)
```

Аналогично можно получить сразу вектор всех значений указанной функции при разных значениях аргумента:

```
» z = 1:0.1:1.8;
```

```
» m = F1(z, 1)
```

Чтобы получить информацию о созданной процедуре, достаточно набрать в командном окне команду:

```
» help F1
```

Пример 2. Построим график двух функций:

**$y1 = 200 \sin(x)/x$ ;  $y2 = x^2$ .**

Для этого создадим М-файл:

```
function y = myfun(x)
```

```
% Вычисление двух функций
```

```
%  $y(1) = 200 \sin(x)/x$ ,  $y(2) = x^2$ .
```

```
y(:,1) = 200*sin(x) ./ x;
```

```
y(:,2) = x.^2;
```

Теперь построим графики этих функций:

```
>> fplot('myfun', [-20 20]), grid
```

Пример 3.

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

В этом случае удобно объединить совокупность коэффициентов  $k$  в единый вектор  $K$ :

$$K = [k1 \ k2 \ k3 \ k4 \ k5]$$

и создать такой М-файл:

```
function y = myfun2(x, K)
```

```
% Вычисление функции
```

```
% y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5))
```

```
y = K(1)+K(2)*x+K(3)*sin(K(4)*x+K(5));
```

Тогда расчет значений этой функции:

```
» K = [1 5 7 2 9];
```

```
» t = 0:1:10;
```

```
» fi = myfun2(t, K)
```



## 6.4. Создание **Script-файлов**

### 6.4.1. Основные особенности **Script-файлов**

- ❑ **Script-файлы** являются независимо (самостоятельно) исполняемыми блоками операторов и команд;
- ❑ все используемые переменные образуют так называемое *рабочее пространство*, которое является общим для всех исполняемых **Script-файлов**;
- ❑ в них отсутствует заголовок, т. е. первая строка определенного вида и назначения;
- ❑ обращение к ним не требует указания никаких имен переменных: все переменные формируются в результате выполнения программы либо сформированы ранее и существуют в рабочем пространстве.

## 6.4.2. Ввод и вывод информации в диалоговом режиме

Для обеспечения взаимодействия с пользователем в процессе выполнения М-файла в системе MatLAB используются такие команды:


***disp*** – осуществляет вывод значений указанной переменной или указанного текста в командное окно. Обращение к ней имеет вид:

***disp*** (<переменная или текст в апострофах>)

Особенностью этой команды является то, что аргумент у нее может быть только один.

Поэтому невозможно без специальных мер осуществить вывод нескольких переменных и, в особенности, объединение текста с численными значениям и некоторых переменных.

Для устранения этого недостатка используют несколько способов:



1. Если все выводимые переменные имеют один и тот же формат (числовой или текстовый), то чтобы вывести значения нескольких переменных в одну строку необходимо объединить соответствующие переменные в вектор.

***disp*** (*[x1 x2 ... x]*)

Пример:

```
» x1=1.24; x2=-3.45; x3=5.76; x4=-8.07;
```

```
» disp([x1 x2 x3 x4])
```

```
1.2400 -3.4500 5.7600 -8.0700
```

Либо для текстовых переменных, например:

```
» x1='psi'; x2='fi'; x3='teta'; x4='w1';
```

```
» disp([x1 x2 x3 x4])
```

```
psi fi teta w1
```

2. Если все выводимые переменные имеют разный формат, то чтобы вывести значения нескольких переменных в одну строку необходимо использовать функцию ***sprintf***. Обращение к ней имеет вид:

$Y = \mathbf{sprintf} ('<текст1> \%g <текст2>', X).$

В результате получается текстовая строка  $Y$ , состоящая из текста, указанного в  $<текст1>$ , и значения числовой переменной  $X$  в соответствии с форматом  $\%g$ , причем текст из фрагмента  $<текст2>$  располагается после значения переменной  $X$ .

Эту функцию можно использовать в команде ***disp*** в виде:


$\mathbf{disp (sprintf ('<текст> \%g', X) )}.$

Пример:

```
» x = -9.30876e-015;
```

```
» disp(sprintf('Параметр1 = %g Volt', x))
```

```
Параметр1 = -9.30876e-015 Volt
```



***input*** – функция ввода информации с клавиатуры в диалоговом режиме

Синтаксис:

$x = \mathit{input}(\text{'<приглашение>'})$

При этом, выполнение операторов программы прекращается. ПК переходит в режим ожидания окончания ввода информации с клавиатуры. После окончания ввода с клавиатуры введенная информация запоминается в программе под именем "x", и выполнение программы продолжается.

***menu*** – создает текущее окно меню пользователя.

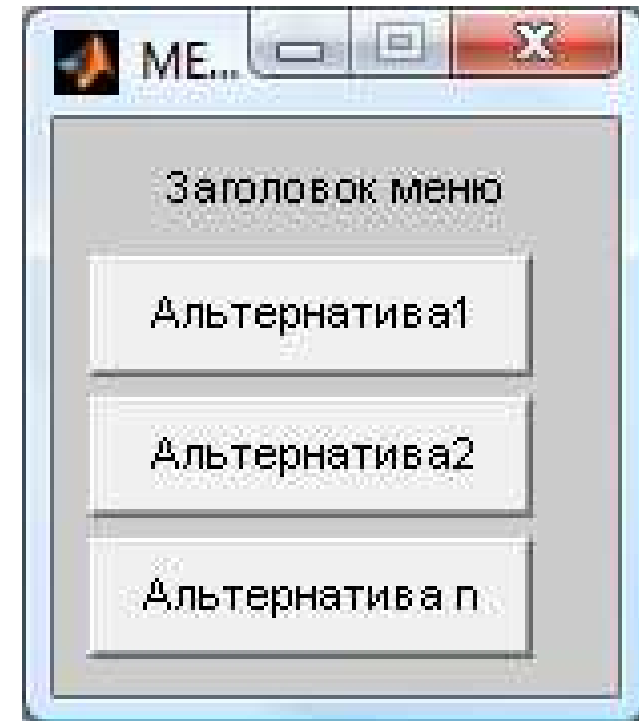
Синтаксис:


$k = \mathit{menu}(\text{'Заголовок меню'}, \text{'Альтернатива1'}, \text{'Альтернатива2'}, \dots, \text{'Альтернатива n'})$

Такое обращение приводит к появлению на экране меню:



Выполнение программы временно приостанавливается, и система ожидает выбора одной из кнопок меню с альтернативами. После правильного ответа исходному параметру "k" присваивается значение номера избранной альтернативы (1, 2, ..., n). В общем случае число альтернатив может быть до 32.





***pause*** – временно прекращает выполнение программы до тех пор, пока пользователь не нажмет любую клавишу клавиатуры. Если после названия команды указать в скобках некоторое положительное целое число *n*, то задержка выполнения программы будет осуществлена на протяжении *n секунд*.

***keyboard*** – выполнение М-файла прекращается, и управление передается клавиатуре. Этот специальный режим работы сопровождается появлением в командном окне MatLAB нового вида приглашения к действиям

*k>>*

В этом режиме пользователь может осуществить любые действия, проверить или изменить данные. При этом ему доступны все команды и процедуры системы MatLAB. Для завершения работы в этом режиме необходимо ввести команду ***return***. Тогда система продолжит работу программы с оператора, следующего за командой ***keyboard***.

## 6.4.3. Операторы управления вычислительным процессом

К операторам управления вычислительным процессом относят: операторы условного перехода и операторы организации циклических процессов.

Все операторы цикла и условного перехода построены в MatLAB в виде составного оператора, который начинается одним из служебных слов ***if***, ***while***, ***switch*** или ***for*** и заканчивается служебным словом ***end***.

### Оператор условного перехода

Конструкция:

```
if <условие>  
    <операторы1>  
else  
    <операторы2>  
end
```




Сокращенная форма условного оператора имеет вид:

```
if <условие>  
    <операторы>
```

Допустима еще одна конструкция оператора условного перехода:

```
if <условие1>  
    <операторы1>  
elseif <условие2>  
    <операторы2>  
elseif <условие3>  
    <операторы3>  
    ...  
else  
    <операторы>  
end
```

Оператор **elseif** выполняется тогда, когда <условие1> не выполнено.



В качестве условия используются выражения типа:  
<имя переменной1> <операция сравнения> <имя переменной2>

Операции сравнения в языке MatLAB:

< - меньше;

> - больше;

<= - меньше или равно;

>= - больше или равно;

= = - равно;

~ = - не равно.

Условие может состоять из нескольких простых условий, объединенных знаками логических операций.

Знаки логических операций:

& - логическая операция "И" ("AND");

| - логическая операция "ИЛИ" ("OR");

~ - логическая операция "НЕТ" ("NOT").



## Оператор переключения

Структура:

```
switch <выражение, скаляр или строка символов>  
case <значение1>  
    <операторы1>  
case <значение2>  
    <операторы2>  
  
...  
otherwise  
    <операторы>  
end
```

Он осуществляет разветвление вычислений в зависимости от значений некоторой переменной или выражения, сравнивая значение, полученное в результате вычисления выражения в строке **switch**, со значениями, указанными в строках со словом **case**. Если значение выражения не совпадает ни с одним из значений в группах **case**, выполняются операторы, которые следуют за словом **otherwise**.

## Операторы цикла

В языке MatLAB есть две разновидности операторов цикла – *условный* и *арифметический*.

Оператор цикла с предусловием имеет вид:

```
while <условие>  
    <операторы>
```

```
end
```

Операторы внутри цикла выполняются лишь в случае, если выполнено условие, записанное после слова **while**. При этом среди операторов внутри цикла обязательно должны быть такие, которые изменяют значения одной из переменных, указанных в условии цикла.

Пример вычисления значений синуса от 0.2 до 4 с шагом 0.2:

```
» i = 1;  
» while i <= 20  
x = i/5;  
si = sin(x);  
disp([x, si])  
i = i+1;  
end
```

Арифметический оператор цикла имеет вид:

**for** <имя> = <НЗ> : <Ш> : <КЗ>

<операторы>

**end**

где <имя> - имя управляющей переменной цикла ("счетчика" цикла);  
<НЗ> - заданное начальное значение этой переменной; <Ш> - значение шага; <КЗ> - конечное значение переменной цикла. Если параметр <Ш> не указан, по умолчанию его значение принимается равным единице.

Чтобы досрочно выйти из цикла (например, при выполнении некоторого условия) применяют оператор **break**. Когда программа сталкивается с этим оператором, выполнение цикла досрочно прекращается, и начинает выполняться оператор, следующий за словом **end**.

Для примера используем предыдущую задачу:

```
» a = ['i', 'x', 'sin(x)'];  
» for i = 1:20  
    x = i/5;  
    si = sin(x);  
    if i==1  
        disp(a)  
    end  
    disp([i,x,si])  
end
```

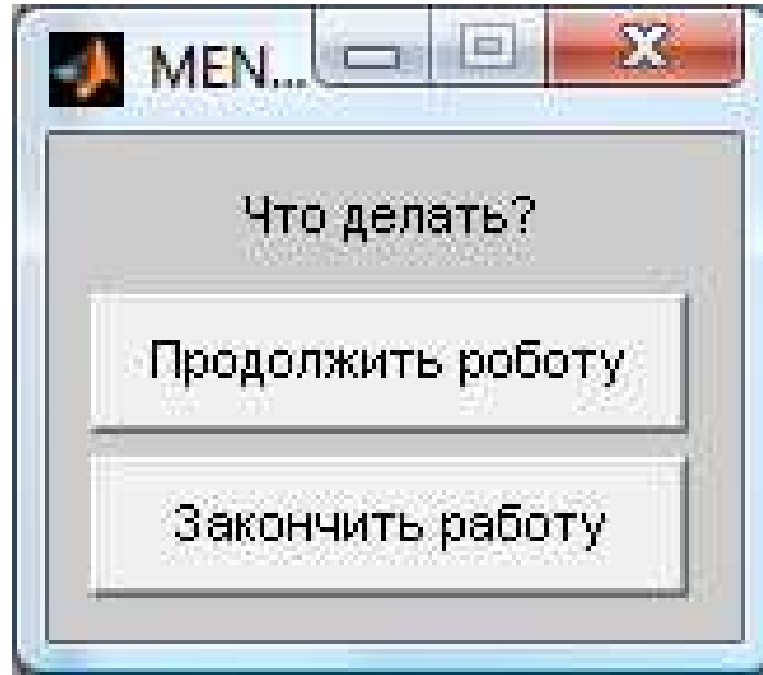
## 6.4.4. Организация повторения действий в *Script-файлах*

Одной из главных задач создания самостоятельной программы есть обеспечение возвращения к началу программы с целью продолжения ее выполнения при новых значениях исходных данных.

Пусть основные операторы созданной программы расположены в *Script-файле* с именем "Yadro.m". Тогда схема обеспечения возврата к началу выполнения этого *Script-файла* может быть, например, такой:

```
k=1;
while k==1
    Yadro
k = menu('Что делать?', 'Продолжить работу',
        'Закончить работу');
end
```

Тогда, после первого выполнения *Script-файла* "Yadro.m" на экране появится окно меню



При нажатии кнопки первой альтернативы значение  $k$  останется равным 1, цикл повторится, а при нажатии второй кнопки  $k$  станет равным 2, цикл закончится и программа перейдет к окончанию работы.