

# *Кафедра ИУ8*

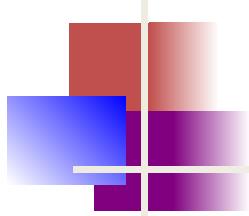
## **«Информационная безопасность»**

### **Учебная практика:**

Основы программирования в среде Matlab

1. Лабораторная работа №1.
2. Лабораторная работа №2
3. Лабораторная работа №3.
4. Лабораторная работа №4
5. Лабораторная работа №5

Выполняются в соответствии с указаниями  
«Учебная практика по курсу MATLAB» (matlab.rar).  
По результатам - итоговая оценка.



# *Кафедра ИУ8*

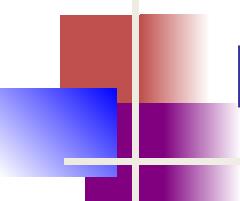
## **«Информационная безопасность»**

### **Лекция:**

Основы программирования в среде Matlab

### **Учебные вопросы:**

1. Основные средства программирования.
2. Основные типы данных.
3. М-файлы сценариев и функций
4. Управляющие структуры



## Рекомендуемая литература:

1. **Дьяконов В.П. MATLAB 6.5 SP1/7.0 + Simulink 5/6. Основы применения.**  
**Серия «Библиотека профессионала».** – М.: СОЛОН. Пресс, 2005. – 800 с.
2. **Курбатова Е.А. MATLAB 7. Самоучитель.** Изд-во: Вильямс. 2005г. 256 с.
3. Иглин С.П. Математические расчеты на базе Matlab. Изд-во "ВНВ-Санкт-Петербург" 2005г. 640 стр.
4. Сергиенко А. Цифровая обработка сигналов (второе издание). СПб, Питер, 2006. - 751 с.
5. Поршнев С.В. MATLAB 7. Основы работы и программирования. Учебник. Изд-во "Бином. Лаборатория знаний" 2006г. 320 стр.
6. <http://matlab.exponenta.ru>
7. **Половко А.М., Бутусов П.Н. MATLAB для студентов.** – СПб.: БХВ-Петербург, 2005. – 320 с.
8. **Потемкин В. Г. Система инженерных и научных расчетов MATLAB 5.x: в 2-х т.**
9. Чен К., Джиблин П. Ирвинг А. MATLAB в математических исследованиях: Пер. с англ. – М.: Мир, 2001. – 346 с.
10. **Ануфриев И.Е., Смирнов А.Б., Смирнова Е.Н. MATLAB 7.** – СПб.: БХВ-Петербург, 2005. – 1104 с.

# Введение

Использование **командного режима** (режима командной строки, командное окно) не является основным при использовании возможностей среды **MATLAB**. Однако при решении ряда серьезных задач возникает необходимость сохранения используемых последовательностей вычислений, а также их дальнейшей модификации. Иными словами, существует необходимость **программирования** решения задач.

Практически невозможно предусмотреть в одной, даже самой большой и мощной, математической системе возможность решения всех задач, которые могут интересовать пользователя. Программирование в системе MATLAB является эффективным средством ее расширения и адаптации к решению специфических проблем. Оно реализуется с помощью языка программирования системы. Программы на языке программирования **MATLAB** сохраняются в виде **текстовых m-файлов**. При этом могут сохраняться как целые программы в виде **файлов-сценариев**, так и отдельные **программные модули — функции**.

Кроме того, важно, что программа может менять структуру алгоритмов вычислений в зависимости от входных данных и данных, создаваемых в ходе вычислений.

С позиций программиста язык программирования системы является типичным **проблемно-ориентированным** языком программирования высокого уровня. Точнее говоря, это даже язык **сверхвысокого** уровня, содержащий сложные операторы и функции, реализация которых на обычных языках (например, Бейсике, Паскале или Си) потребовала бы много усилий и времени. К таким функциям относятся **матричные функции, функции быстрого преобразования Фурье (БПФ) и др., а к операторам — операторы построения разнообразных графиков, генерации матриц определенного вида и т. д.**

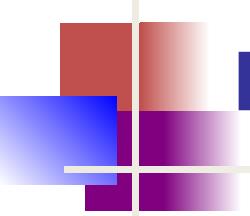
# 1. Основные средства программирования

Программами в системе *MATLAB* являются **т-файлы** текстового формата, содержащие запись программ в виде программных кодов.

Язык программирования системы *MATLAB* имеет следующие средства:

- ❖ данные различного типа;
- ❖ константы и переменные;
- ❖ операторы, включая операторы математических выражений;
- ❖ встроенные команды и функции;
- ❖ функции пользователя;
- ❖ управляющие структуры;
- ❖ системные операторы и функции;
- ❖ средства расширения языка.

# Основные средства программирования



Коды программ в системе **MATLAB** пишутся на языке высокого уровня, достаточно понятном для пользователей умеренной квалификации в области программирования. Язык программирования **MATLAB** является типичным *интерпретатором*. Это означает, что каждая инструкция программы распознается и тут же исполняется, что облегчает обеспечение **диалогового режима общения** с системой. Этап компиляции всех инструкций, т. е. полной программы, отсутствует. Высокая скорость выполнения программ обеспечена наличием заведомо откомпилированного ядра, хранящего в себе критичные к скорости выполнения инструкции, такие как базовые математические и иные функции, а также тщательной отработкой системы контроля синтаксиса программ в режиме интерпретации.

Интерпретация означает, что **MATLAB** не создает исполняемых конечных программ. Они существуют лишь в виде m-файлов. Для выполнения программ необходима среда **MATLAB**. Однако для программ на языке **MATLAB** созданы компиляторы, транслирующие программы **MATLAB** в коды языков программирования С и С++. Это решает задачу создания исполняемых программ, первоначально разрабатываемых в среде MATLAB.

## 2. Основные типы данных

Типы данных `array` и `numeric` являются *виртуальными* («кажущимися»), поскольку к ним нельзя отнести какие-либо переменные. Они служат для определения и комплектования некоторых типов данных. Таким образом, в **MATLAB** определены следующие основные типы данных, в общем случае представляющих собой многомерные массивы:

`single` — числовые массивы с числами одинарной точности;

`double` — числовые массивы с числами удвоенной точности;

`char` — строчные массивы с элементами-символами;

`sparse` — наследует свойства `double`, разреженные матрицы с элементами-числами удвоенной точности;

`cell` — массивы ячеек; ячейки, в свою очередь, тоже могут быть массивами;

`struct` — массивы структур с полями, которые также могут содержать массивы;

`function_handle` — дескрипторы функций:

`int32, uint32` — массивы 32-разрядных чисел со знаком и без знаков;

`intl6, uint!6` — массивы 16-разрядных целых чисел со знаком и без знаков;

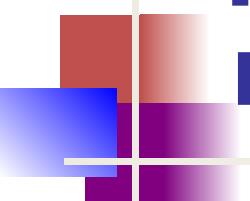
`int8, uint8` — массивы 8-разрядных целых чисел со знаками и без знаков.

### 3. Виды программирования

Система программирования MATLAB позиционируется как язык высокого уровня для научно-технических расчетов. Язык программирования системы MATLAB вобрал в себя все средства, необходимые для реализации различных видов программирования:

- процедурного;
- операторного;
- функционального;
- логического;
- структурного (модульного);
- объектно-ориентированного;
- визуально-ориентированного.

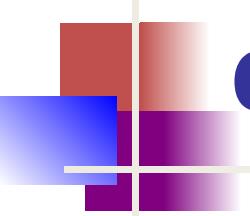
# Двойственность операторов, команд и функций



Для языка системы **MATLAB** различие между командами (выполняемыми при вводе с клавиатуры) и программными операторами (выполняемыми из программы) является условным. И команды, и программные операторы могут выполняться как из программы, так и в режиме прямых вычислений. Под **командами** далее в основном понимаются средства, управляющие периферийным оборудованием, под **операторами** — средства, выполняющие операции с operandами (данными).

**Функция** преобразует одни данные в другие. Для многих функций характерен возврат значений в ответ на обращение к ним с указанием списка входных параметров — **аргументов**. Например, говорят, что функция  $\sin(x)$  в ответ на обращение к ней возвращает значение синуса аргумента  $x$ . Поэтому функцию можно использовать в арифметических выражениях, например  $2*\sin(x+1)$ . Для операторов (и команд), не возвращающих значения, такое применение обычно абсурдно.

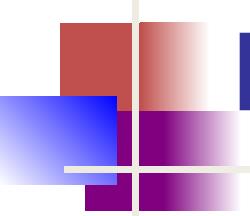
# 4. М-файлы сценариев и функций



И в командной строке, и в текстах м-файлов функции записываются только **малыми** буквами. Для функций, возвращающих ряд значений или массивов (например X, Y, Z,...), запись имеет следующий вид:

[X,Y,Z, ...]=f\_name(Список\_параметров)

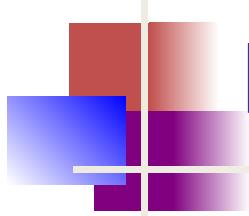
Есть два типа м-файлов: **файлы-сценарии** и **файлы-функции**. В процессе своего создания они проходят синтаксический контроль с помощью встроенного в систему MATLAB редактора/отладчика м-файлов.



# М-файлы сценариев

**Файл-сценарий**, именуемый также Script-файлом, является просто записью серии команд без входных и выходных параметров. Важны следующие свойства файлов-сценариев:

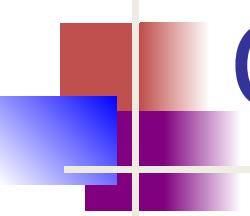
- ❖ они не имеют входных и выходных аргументов;
- ❖ работают с данными из рабочей области;
- ❖ в процессе выполнения не компилируются;
- ❖ представляют собой зафиксированную в виде файла последовательность операций, полностью аналогичную той, что используется в сессии.



# Пример файла-сценария

```
%Plot with color red  
%Строит график синусоиды линией красного цвета  
%с выведенной масштабной сеткой в интервале [xmin,xmax]  
x=xmin:0.1:xmax;  
plot(x, sin(x),'r')  
grid on
```

Первые три строки здесь — это комментарий, остальные — тело файла.



# Структура файла-функции

**М-файл-функция** является типичным объектом языка программирования системы **MATLAB**. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует аппарат локальных переменных.

Структура такого модуля с одним выходным параметром выглядит следующим образом:

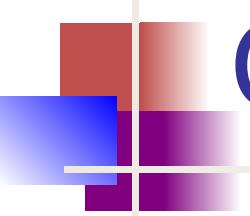
```
function var=f_name(Список_параметров)
%Основной комментарий
%Dополнительный комментарий
Тело файла с любыми выражениями
var=выражение
```

# Структура файла-функции

М-файл-функция имеет следующие свойства:

- он начинается с объявления `function`, после которого указывается имя переменной `var` — выходного параметра, имя самой функции и список ее входных параметров;
- функция возвращает свое значение и может использоваться в виде `name (Список_параметров)` в математических выражениях;
- все переменные, имеющиеся в теле файла-функции, являются *локальными*, т. е. действуют только в пределах тела функции;
- файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;
- правила вывода комментариев те же, что у файлов-сценариев;
- файл-функция служит средством расширения системы `MATLAB`;
- при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы `MATLAB`.

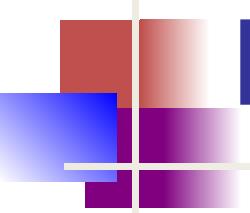
Последняя конструкция `var=выражение` вводится, если требуется, чтобы функция возвращала результат вычислений.



# Структура файла-функции

Если выходных параметров больше, то они указываются в квадратных скобках после слова `function`. При этом структура модуля имеет следующий вид:

```
function [var1,var2....]=f_name(Список_параметров)
%Основной комментарий
%Dополнительный комментарий
Тело файла с любыми выражениями
var1=выражение
var2=выражение
```



# Примеры файл-функции

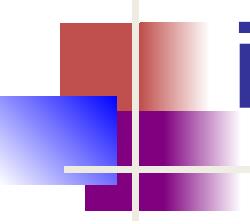
```
function y=Norm(x,m,D);
sko=sqrt(D);
y=exp((-x-m).^2)/(2*D))
/(sko*sqrt(2*pi));
```

Использование файл-функции в М-файле

```
figure(1);
x=1:70; m1=25; D1=20; m2=40; D2=30;
y1=Norm(x,m1,D1);
y2=Norm(x,m2,D2);
y3=y1+y2;
plot(x,y1,x,y2,x,y3); grid on;
```

## 5. Управляющие структуры

Помимо программ с *линейной структурой*, инструкции которых исполняются строго по порядку, существует множество программ, структура которых *нелинейна*. При этом ветви программ могут выполняться в зависимости от определенных условий, иногда с конечным числом повторений — циклов, иногда в виде циклов, завершаемых при выполнении заданного условия. Практически любая серьезная программа имеет нелинейную структуру. Для создания таких программ необходимы специальные управляющие структуры. Они имеются в любом языке программирования, и в частности в MATLAB.



# if-else-end

**if** Условие Выражение\_1 Оператор\_отношения Выражение\_2

Инструкции\_1

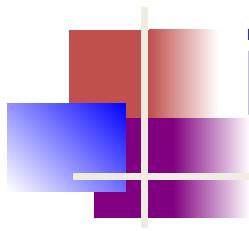
**else**

Инструкции\_2

**end**

В качестве Операторов\_отношения используются  
следующие операторы: **==, <, >, <=, >= ИЛИ ~=.**

Все эти операторы представляют собой пары  
символов без пробелов между ними.

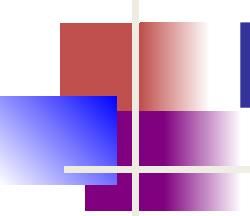


# if-elseif-else-end

- if Условие
- Инструкции\_1
- elseif Условие
- Инструкции\_2
- else
- Инструкции\_3
- end

## ПРИМЕР

```
if x<5  y=0.1;  
elseif x<10 y=0.5;  
elseif x<15 y=0.75;  
else  y=1.0;  
end;
```

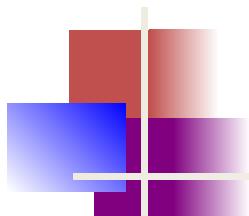


# Циклы for...end

Конструкции циклов типа `for...end` обычно используются для организации вычислений с заданным числом повторяющихся циклов. Конструкция такого цикла имеет следующий вид:

```
for var=Выражение  
Инструкция....  
end
```

Выражение чаще всего записывается в виде  $s:d:e$ , где  $s$  — начальное значение переменной цикла  $var$ ,  $d$  — приращение этой переменной и  $e$  — конечное значение управляющей переменной, при достижении которого цикл завершается. Возможна и запись в виде  $s :e$  (в этом случае  $d=1$ ). Список выполняемых в цикле инструкций завершается оператором `end`.

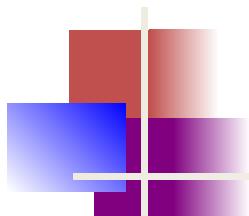


# Пример (for...end )

```
figure(1)
for x=1:100
    y(x)=0.01*x;
end
plot(1:x,y(1:x)); grid on;
```

# Пример совместного использования операторов for-end и if-end

```
figure(2)
x=1:0.5:100;
for i=1:length(x)
    if (x(i)>0)&&(x(i)<20)
        y(i)=sin(x(i));
    end
    if (x(i)>=20)&&(x(i)<50)
        y(i)=2+cos(x(i));
    end
    if (x(i)>=50)&&(x(i)<=100)
        y(i)=log(x(i));
    end
end
plot(x,y); grid on;
```



# Циклы while...end

Цикл типа `while` выполняется до тех пор, пока выполняется Условие:

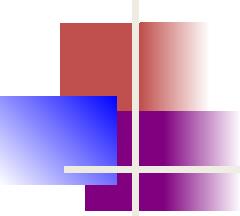
`while` Условие

Инструкции

`end`

# Пример (while, if-end)

```
%-----while-----
x=0; dx=0.01; i=1;
while x<=40,
    if x<5    y=0.1;
    elseif x<10 y=0.5;
    elseif x<15 y=0.75;
    else    y=1.0;
    end;
    x=x+dx;
    x1(i)=x;
    y1(i)=y;
    i=i+1;
end
figure(1)
plot(x1,y1); grid on;
```



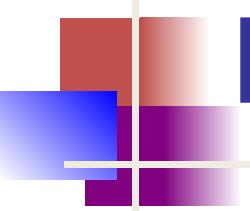
## Для осуществления множественного выбора (или ветвления) используется конструкция с переключателем типа switch

```
switch switch_Выражение
    case case_Выражение
        Список_инструкций
    case {case_Выражение1, case_Выражение2, ....}
        Список_инструкций
    ...
    otherwise. Список_инструкций
end
```

Если выражение после заголовка `switch` имеет значение одного из выражений `case_Выражение...`, то выполняется блок операторов `case`, в противном случае — список инструкций после оператора `otherwise`. При выполнении блока `case` исполняются те списки инструкций, для которых `case_Выражение` совпадает со `switch_Выражением`.

# Пример использования оператора switch-case-end

```
figure(3)
z=zeros(1,10);
for x=1:10
    switch x
        case {1,2,3}
            z(x)=4;
        case {4,5,6,7}
            z(x)=2;
        otherwise
            z(x)=7;
    end
end
bar(z);
```



## Выводы

В среде **MATLAB** заложены достаточно мощные и удобные средства программирования.

Программирование в среде **MATLAB** основано на использовании **М-файлов-сценариев** и **М-файлов-функций**.

В среде **MATLAB** заложена возможность поддержки функций с переменным числом аргументов.

Использование управляющих структур (условные операторы, циклы) и функции диалога позволяет организовывать нелинейные структуры программ.