MATLAB в научных исследованиях

Лекция № 3. Конструкции языка. Типы данных. Функции пользователя

Н. Ю. Золотых

ННГУ, ВМК, сентябрь–декабрь 2008

Содержание

1. Управляющие конструкции

- Команда условия `if`
- Команда цикла `while`
- Команда цикла с параметром `for`
- Команда выбора `switch`

1.1. Команда `if`

```
if <условие1>
    <команды>
elseif <условие2>
    <команды>
elseif <условие3>
    <команды>
...
else
    <команды>
end
```

`<условие>` можно получить в результате

- операций-отношений: `< > == <= >=`,
- в результате логических операций: `~= & | ~ && ||`,
- специальных функций: `any`, `all` и др.

Если `V` — массив, то условие `V` эквивалентно условию `all(V)`.

Примеры:

```
a = [0 1 Inf NaN];
b = [2 0 0 2];
~a
~b
a | b
a & b

a = [1 2];
b = [2 1];
a == b
a < b
a > b
any(a < b)
all(a < b)
```

Пример с командой `if`

$$f(x) = \begin{cases} 0, & \text{если } x < 0, \\ x, & \text{если } 0 \le x < 1, \\ \sqrt{x}, & \text{если } x \ge 1. \end{cases}$$

Две реализации:

```
if x < 0
    f = 0;
else
    if x < 1
        f = x;
    else
        f = sqrt(x);
    end
end
```

```
if x < 0
    f = 0;
elseif x < 1
    f = x;
else
    f = sqrt(x);
end
```

1.2. Команда `while`

```
while <выражение>
    <команды>
end
```

Пример

Вычисление константы `eps`

```
e = 1;
while 1 + e ~= 1,
    e = e/2;
end
e = 2*e
```

1.3. Команда `for`

```
for <переменная> = <выражение>
    <команды>
end
```

Пример: матрицы Гильберта

```
n = 10;
H = zeros(n);
for i = 1:n
    for j = 1:n
        H(i, j) = 1/(i + j - 1);
    end
end
```

Для генерации матриц Гильберта есть стандартная функция `hilb(n)`.

Она реализована без единого цикла.

1.4. Команда switch

```
switch <выражение>
case {<список значений 1>}
    <команды>
case {<список значений 2>}
    <команды>
...
otherwise
    <команды>
end
```

Пример

```
switch lower(method)
    case {'linear','bilinear'}
        disp('Method is linear')
    case 'cubic'
        disp('Method is cubic')
    case 'nearest'
        disp('Method is nearest')
    otherwise
        disp('Unknown method')
end
```

2. Типы данных (классы)

- `ARRAY` (плотный или разреженный)
  - `logical`
  - `char`
  - `NUMERIC`
    - `int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`, `uint64`
    - `single`
    - `double`
  - `cell`
  - `structure`
    - пользовательские классы
  - классы `Java`
  - `function handle`

2.1. Массивы структур (`structure arrays`)

Структурой называется абстрактный тип данных, представляющий собой коллекцию значений (*полей*) разных типов, доступ к которым осуществляется по имени (*имени поля*).

Массив структур — это коллекция структур, доступ к которым происходит по индекс

```
S.name = 'Isaac Newton';
S.age = 38;
```

name	*age*
'Isaac Newton'	38

Набор полей структуры может изменяться динамически. Также динамически могут меняться размеры массива структур:

```
S(2).name = 'Blaise Pascal' ;
S(2).age = 23;
```

№	*name*	*age*
1	'Isaac Newton'	38
2	'Blaise Pascal'	23

Другой способ создания структуры:

```
S(3) = struct('name', ...
  'Carl F. Gauss', 'age', 43);
```

№	*name*	*age*
1	'Isaac Newton'	38
2	'Blaise Pascal'	23
3	'Carl F. Gauss'	43

Добавим еще одно поле:

```
S(3).profession = 'mathematician'
```

№	*name*	*age*	*profession*
1	‘Isaac Newton’	38	`[]`
2	‘Blaise Pascal’	23	`[]`
3	‘Carl F. Gauss’	43	‘mathematician’

2.2. Массивы ячеек (cell arrays)

Ячейкой (cell) называется контейнер, который может содержать в себе произвольный из рассмотренных типов данных (т.е. это может быть массив чисел с плавающей запятой, массив символов, массив структур и др.)

Массив ячеек — это коллекция ячеек, доступ к которым происходит по индексу. Таки образом, массив ячеек может объединять разнотипные данные. Массив может быть одномерным, двумерным или многомерным.

Доступ к ячейкам осуществляется указанием после имени массива индекса элемента в фигурных скобках.

```
for n = 1:5
  M{n} = hadamard(2^n);
end
M{2}
```

Другой способ создать массив ячеек — это перечислить его элементы построчно, разделяя элементы в одной строке пробелами или запятыми, а сами строки — точкой запятой или символом перехода на новую строку.

Все элементы должны быть заключены в фигурные скобки.

Например,

```
A = hadamard(4)
C = {A sum(A) prod(A) 'matrix A'}
```

Чтобы создать массив пустых ячеек достаточно воспользоваться функцией `cell` с указанием размеров массива.

```
M = cell(5,1)
```

3. Пользовательские функции

Программа — последовательность команд, записанных в файл (*m-файл*)

Основные принципы

- Программа — обычный текстовый файл. Ее можно создавать, например, в блокноте Windows или во встроенном редакторе, который вызывается командой `edit`.
- Команды записываются так же как и в командном окне и разделяются знаками «`;`» или «`,`» или символами перехода на новую строку.
- Конец программы никак не помечается (кроме программ, содержащих вложенные — nested — функции). Досрочный выход — по команде `return`.
- Символ `%` начинает комментарий.
- Первые строки комментария подключаются в систему справки и вызываются командой `help имя_программы`.

Типы М-файлов

Сценарии (скрипты)	*Функции*
Работают с данными в рабочем пространстве и общими (*глобальными*) данными	Работают со своими собственными (*локальными*) данными и общими (*глобальными*) данными. Могут работать с данными рабочего пространства.
Не имеют входных аргументов и не могут возвращать значения	Могут иметь входные аргументы и могут возвращать значения
Как правило, используются для отладки и прикидочных расчетов	Расширяют возможности языка MATLAB

Данные рабочего пространства

Глобальные данные

Локальные данные функции1

Локальные данные функции2

3.1. Программы-функции (m-функции)

Заголовок функции:

```
function [y1, y2, ..., ym] = ff(x1,x2,...,xn)
function y = ff(x1,x2,...,xn)
function ff(x1, x2, ..., xn)
function [y1, y2, ..., ym] = ff
```

Способы вызова функции:

```
[y1, y2, ..., ym] = ff(x1, x2, ..., xn)
y = ff(x1, x2, ..., xn)
ff(x1, x2, ..., xn)
[y1, y2, ..., ym] = ff
```

3.2. Подчиненные функции

- Подфункции (subfunctions)
- Вложенные функции (nested functions)
- Частные функции (private functions)

3.2.1. Подфункции (subfunctions)

М-файл с программой-функцией может содержать описание нескольких функций. Только первая функция (*основная*) доступна извне. Остальные (*подфункции*) могут быть вызваны из основной функции или из другой подфункции того же самого m-файла.

```
function [...] = name(...)

    a = ...;
    b = ...;

function [...] = name1(...)

    a, b не видны

function [...] = name2(...)

    a, b не видны
```

3.2.2. Вложенные функции (nested functions)

```
function [...] = name(...)

    a = ...;
    b = ...;

    function [...] = name1(...)
       a, b доступны
    end

    function [...] = name2(...)
       a, b доступны
    end

end
```

3.2.3. Частные функции (private functions)

Частные функции — это функции, размещенные в каталоге с именем `private`. Частные функции доступны из функций, расположенных в родительском каталоге. Этот каталог не следует указывать в путях доступа.

3.3. Функции с переменным числом аргументов

`nargin` — фактическое число входных аргументов

`nargout` — фактическое число выходных параметров

3.4. Функции с произвольным числом аргументов

```
function varargout = ff(y)
function y = ff(varargin)
```

`varargin`, `varargout` — массивы ячеек с входными и выходными аргументами соответственно.

3.5. Глобальные переменные

Функция объявления глобальных переменных

```
global a1 a2 ...
```

должна появиться во всех функциях, в которых используются эти переменные.