Тема 4.6

Программирование на языке MATLAB

Вопросы для изучения

4.30 Объектно ориентированные возможности MATLAB
4.31 Запись информации во внешние файлы. Чтение данных из файлов в рабочую область

4.30 Объектно ориентированные возможности MATLAB

MATLAB позволяет реализовать концепцию объектно-ориентированного программирования. В MATLAB можно создавать и манипулировать объектами, которые скрывают реальные данные за контролируемыми методами работы с ними, допускают переопределение операций и дают возможность наследовать их свойства производным объектам-потомкам.

В основе объектно-ориентированного программирования лежат три основных положения:

- инкапсуляция — объединение данных, программ и передача данных через входные и выходные параметры функций. В результате появляется новый элемент программирования — объект.
- наследование — возможность создания родительских объектов и на их основе новых дочерних объектов, наследующих свойства родительских объектов. Возможно также множественное наследование, при котором класс наследует свойства нескольких родительских объектов.
- полиформизм — присвоение некоторому действию одного имени, которое в дальнейшем используется по всей цепочке создаваемых объектов сверху донизу, причем каждый объект выполняет это действие присущим ему способом.

Последний принцип объектно-ориентированного программирования, полиморфизм, в MATLAB не реализован.

Объект можно определить как некоторую структуру, принадлежащую к определенному классу.

Классом в MatLAB принято называть определенную форму представления вычислительных объектов в памяти ЭВМ в совокупности с правилами (процедурами) их преобразования.

Класс определяет тип переменной, а правила - операции и функции, которые могут быть применены к этому типу.

В свою очередь, тип определяет объем памяти, которая отводится записи переменной в память ЭВМ и структуру размещения данных в этом объеме.

Операции и функции, которые могут быть применены к определенному типу переменных, образовывают **методы** этого класса.

Каждый элемент созданный в соответствии с описанием класса принято называть объектом, т. е. класс это описание а объект это то что в соответствии с этим описанием создано.

В системе MatLAB определены шесть встроенных классов вычислительных объектов:

- double - числовые массивы и матрицы действительных или комплексных чисел с плавающей запятой в формате двойной точности;
- sparse - двумерные действительные или комплексные разреженные матрицы;
- char - массивы символов;
- struct - массивы записей (структуры);
- cell - массивы ячеек;
- uint8 - массивы 8-битовых целых чисел без знаков.

Все основные типы MATLAB представляют собой встроенные классы, а переменные- объекты этих классов.

Пользователь имеет возможность вводить свои классы, а также переопределять и доопределять методы всех существующих классов.

Для MATLAB характерно, что никакие классы объектов (в том числе заново создаваемые) не требуют объявления.

Например, создавая переменную

name='Иван',

автоматически получаем объект в виде переменной name класса char.

Таким образом, для переменных MATLAB принадлежность к тому или иному классу определяется их значением.

Является ли переменная объектом, можно определить при помощи функции

isobject(имя переменной).

Для создания нового класса объектов нужно спроектировать структуру MATLAB, которая будет хранить данные, принадлежащие объекту, и определить функции-методы работы с этими объектами.

Для создания новых классов объектов служат конструкторы классов. По существу, это m-файлы, имена которых совпадают с именами классов @Имя_класса, но без символа @. Символом @ помечаются подпапки системы MATLAB, в которых имеются конструкторы классов например в подпапках MATLAB\TOOLBOX.

Язык программирования MATLAB не имеет деклараций, в том числе деклараций новых классов и типов. Поэтому любой объект - представитель некоторого класса создается в момент вызова функции-конструктора этого класса. Следовательно, для создания объекта нужно создать хотя бы один метод (конструктор) в упомянутой папке-контейнере его класса.

Все поля структуры, хранящей данные класса являются скрытыми (private), то есть их поля доступны только из методов данного класса, напрямую в выражениях их использовать нельзя.

Создание класса или объекта

Для создания класса объектов или объектов, а также для их идентификации служит функция class.

Формы применения:

- OBJ=class(struct[ ], 'classname' ,PARENT1, PARENT2....) — создает объект класса 'classname' на базе структуры S и родительских объектов PARENT1, PARENT2,... При этом создаваемый объект наследует структуру и поля родительских объектов. Объекту OBJ в данном случае присуще множественное наследование и он не может иметь никаких полей, кроме унаследованных от родительских объектов.
- class (OBJ) — возвращает класс указанного объекта OBJ.

Для контроля принадлежности заданного объекта к некоторому классу служит функция isa:

isa(OBJ, 'Имя_класса') — возвращает логическую 1, если OBJ принадлежит классу с указанным именем.

Пример:

```
» X=[1 2 3];
» isa(X,'char')
   ans =   0
» isa(X,'double')
   ans =  1
```

Для получения списка методов данного класса объектов используются функции methodsview и methods.

Отличиями от what(‘ имя класса’) является то, что эти функции возвращают информацию также и о классах Java, но информация выводится в отдельном окне, не сообщается информация о папках, все методы из всех папок собраны вместе, и повторяющиеся имена методов удалены:

Methodsview(‘имя класса’) или methods(‘имя класса’,’-full’) — в отдельном окне возвращают полное описание методов класса, включая информацию о наследовании, а для классов Java — и о подписях и атрибутах;

M=methods ('имя класса',' -full ') — возвращает ту же информацию в массиве ячеек M;

M=methods(' имя класса ') — возвращает массив ячеек с перечислением методов, относящихся к заданному классу объектов;

Следующие две функции могут использоваться только внутри конструкторов классов:

inferiorto (' CLASS1'. 'CLASS2'....) и superiortot 'CLASS1', 'CLASS2'....)

Они определяют низший и высший приоритеты классов по отношению к классу конструктора.

which (‘имя метода’) — находит загруженный Java класс и все классы MATLAB, которым принадлежит данный метод;

which (‘-all’, ‘имя метода’) — находит все классы, которым принадлежит данный метод.

Рассмотрим пример конструктора, создающего объекты класса "полином". Этот конструктор должен находиться в файле @polynom/polynom.m.

```
function p = polynom(a)
    %POLYNOM Polynomial class constructor.
    % p = POLYNOM(v)
        if n = = 0
        p.c = [];
        p = class(p, 'polynom');
            elseif isa(a,'polynom')
            p = a;
        else
        p.c = a(:).';
        p = class(p,'polynom');
end
```

Данный конструктор создает полином из заданного вектора, содержащего коэффициенты полинома при убывающих степенях x. Если же в конструктор не передавать никакого аргумента, то будет создан "пустой" полином.

4.31 Запись информации во внешние файлы. Чтение данных из файлов в рабочую область

Создание программ часто предполагает сохранение результатов расчетов в файлы для их дальнейшего анализа, обработки и хранения.

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

Для сохранения и последующей загрузки каких-либо данных в MatLab предусмотрены две функции:

- сохранение данных

save <имя файла> <имена переменных>

- загрузка данных

load <имя файла> <имена переменных>

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.

Пример:

```
function save_load
x = ones(5);
y = 5;
s = 'hello';
save params x y s;
x = zeros(5);
y = 0;
s = ' ';
load params x y s;
disp(x);
disp(y);
disp(s);
```

В данной программе сначала выполняется инициализация переменных x, y, s, затем, они сохраняются в файл params.mat, заменяются другими значениями и после загрузки отображаются начальные значения.

Недостатком функций save и load является то, что они работают с mat-файлами и не позволяют загружать или сохранять данные в других форматах.

Для загрузки информации из файлов, созданных другими программными продуктами для дальнейшей обработки результатов в MatLab предусмотрены 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 получить идентификатор файла, и открыть его используется функция fopen

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

где <режим работы> - может принимать значения, приведенные в табл. 4.73.

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

Таблица 4.73. Режимы работы с файлами в MatLab

Параметр <режим работы>	Описание
‘r'	чтение
'w'	запись (очищает 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.

Пример программы записи и считывания данных из бинарного файла:

```
A = [1 2 3 4 5];
fid = fopen('my_file.dat', 'wb');    % создание переменной и 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
fclose(fid);                         % закрытие файла
```

В результате работы данной программы в рабочем каталоге будет создан файл my_file.dat размером 40 байт, в котором будут содержаться 5 значений типа double, записанных в виде последовательности байт (по 8 байт на каждое значение). Функция fread() считывает последовательно сохраненные байты и автоматически преобразовывает их к типу double, т.е. каждые 8 байт интерпретируются как одно значение типа double.

Если общее количество элементов файла неизвестно, либо оно меняется в процессе работы программы данные из файла необходимо считывать до тех пор, пока не будет достигнут его конец.

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

```
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
```

В данной программе динамически формируется вектор-строка по мере считывания элементов из входного файла. MatLab автоматически увеличивает размерность векторов, если индекс следующего элемента на 1 больше максимального.

Функция fread() записана с двумя выходными параметрами V и N. Первый параметр содержит значение считанного элемента, а второй – число считанных элементов. В данном случае значение N будет равно 1 каждый раз при корректном считывании информации из файла, и 0 при считывании служебного символа EOF, означающий конец файла.

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

Пример

```
fid = fopen('my_file.dat', 'wb');
str='Привет MatLab';                    % строка для записи
fwrite(fid, str, 'int16');              % запись в файл
fclose(fid);
fid = fopen('my_file.dat', 'rb');
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);
```

Выходными результатами многих программ являются текстовые файлы, в которых явным образом записаны числа или текст, например, при экспорте данных из Excel числа могут быть записаны в столбик и разделены запятой

Для этих целей были разработаны функции чтения

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

и записи

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

где 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.

Пример записи матрицы Y состоящей из чисел в файл, в котором числовые значения должны разделяться точкой с запятой.

```
fid = fopen('mye1.txt', 'w');
if fid == -1
    error('File is not opened');
end
fprintf(fid, '%6d %.4f %.4f %.4f %.4f %d\r\n', y');
fclose(fid);
```

Переменная Y имеет знак транспонирования ‘, т.к. данные в файл записываются по столбцам матрицы.

Перед спецификаторами стоят числа, которые указывают сколько значащих цифр числа должно быть записано в файл: %6d говорит о том, что целые числа должны иметь 6 значащих цифр, %.4f означает, что после запятой будет отображено только 4 цифры.

В форматной строке были использованы управляющие символы:

\r – возврат каретки;

\n – переход на новую строку

которые необходимы для формирования строк в выходном файле.

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

```
fid = fopen('my_excel.txt', 'r');
S = fscanf(fid, '%f');
fclose(fid);
```

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

Чтобы сохранить верный формат данных матрицы из n столбцов и m строк функцию fscanf() следует записать

```
S = fscanf(fid, ‘%f, ', [n m]);
```

тогда на выходе получится матрица S размером в n столбцов и m строк с соответствующими числовыми значениями.

Форматная строка в виде '%d,%f,%f,%f,%f,%d' и состоит из спецификаторов:
%d – работа с целочисленными значениями;
%f – работа с вещественными значениями.

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. 4.74.

Таблица 4.74 - Список основных спецификаторов для функций fscanf() и fprintf()

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

С помощью функции fprintf() можно записать значения двух и более переменных разного формата.

```
str = 'Hello';
y = 10;
count = fprintf(fid, '%d\r\n%s\r\n', y, str);
```

содержимое файла будет иметь вид:

```
10
Hello
```

Таким образом можно осуществлять запись разнородных данных в файл в требуемом формате.

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)
```

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

н байт на пиксел.

Пример загрузки растрового изображения

```
[A, map]=imread('1024.bmp','bmp');
```

где A – матрица размером 1024x1024xN точек;
map – цветовая карта загруженного изображения.

Значение N показывает число байт, расходуемых на представление 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');
```

В результате в рабочем каталоге MatLab будет сохранено изображение в формате bmp с исходной цветовой картой.