Programación en lenguaje MATLAB

Clase 3

Dr. Ing. Rodrigo Gonzalez
rodralez@frm.utn.edu.ar

Universidad Tecnológica Nacional,
Facultad Regional Mendoza.

Resumen

1. Funciones
   - Estructura
   - Visibilidad de las variables
   - Comparación entre programa y función
2. Importar datos
   - Importar/exportar desde el Workspace
   - Otras formas de importar datos
   - Lectura de archivos
3. Exportar datos
   - Otras formas de exportar datos
   - Escritura de datos en un archivo
4. Funciones para el manejo de cadena de caracteres
   - Función `sprintf()`
   - Función `sscanf()`
5. Herramientas estadísticas
   - Funciones aleatorias
   - Funciones estadísticas

Definición

- Funciones de MATLAB (built-in) o definida por el usuario (user-defined).
- En matemática se define función como $y = f(x)$.
- En MATLAB es similar:

```
function[oa1,oa2,...]  = nombre_fun
(ia1,ia2,...)
```

- Los nombres de función cumplen las mismas reglas de los nombre de variables.
- Se crea un archivo *.m. **Debe tener el mismo nombre que la función**.

Declaración

```
function[oa1,oa2,...]  = nombre_fun (ia1,ia2,...)
```

- `function` siempre en minúsculas.
- `nombre_fun` nombre de la función.
- Argumentos de entrada se definen entre paréntesis, separados por comas. Cualquier tipo de variable (number, char, etc.).
- Una función puede no tener argumentos de entrada (cuasi programa).
- Argumentos de salida entre corchetes, separados por comas. Cualquier tipo de variable (number, char, etc.).
- Si tiene solo un argumento de salida, se pueden omitir corchetes.
- Una función puede no tener argumentos de salida.
- Debe haber coherencia entre:
  - datos suministrados y dimensiones de argumentos de entrada.
  - dimensiones de argumentos de salida y datos esperados.

Sección ayuda

```
function [oa1,oa2] = nombre_fun (ia1,ia2)
% Línea H1
% Texto de la ayuda
% de la función.
 Cuerpo de la función
 end
```

- La sección ayuda son líneas comentadas seguidas a la declaración de la función. Es opcional.
- La línea H1 incluye primero el nombre de la función. A continuación, el objetivo de la función. Se incluye para usarse con el comando `lookfor`.
- El texto contiguo a H1 es la ayuda de la función. Se usa con el comando `help` o tecla F1.
- **`end`** indica el final de la función. Es opcional aunque conviene su uso.

Funciones

Ejercicio 1

Cree la función `int_compuesto` que calcule el interés compuesto, según la ecuación,

$$C_F = C_I(1 + r)^n$$

Donde:

- C_I es el capital inicial.
- C_F es el capital final.
- r es el interés por periodo, entre 0 y 1.
- n es la cantidad de periodos.

Prototipo de la función:

```
function Cf = int_compuesto (Ci, r, n)
```

Funciones

Ejercicio 2

- Verifique que la función `int_compuesto` opera con vectorización.
- Cree el programa `prueba_int_compuesto` que calcule el capital final para dos escenarios: un depósito anual y 12 depósitos mensuales.
- La función debe recibir 3 vectores del mismo orden: capitales iniciales, intereses anuales y periodos.
- Capital inicial de $1000 y tasa anual del 12%.

$$C_F = C_I(1 + r)^n$$

```
function Cf = int_compuesto (Ci, r, n)
```

- Agregue la línea H1. En consola ejecute `lookfor int_compuesto`.
- A continuación de H1, agregue una pequeña sección ayuda. En consola ejecute `help int_compuesto`.

Visibilidad de las variables

- Según su visibilidad, las variables se dividen en locales y globales.
- Todas las variables utilizadas dentro de una función son locales. No tienen visibilidad fuera de la función.
- Al salir de la función estas variables desaparecen del Workspace.
- Sin embargo, es posible hacer que una variable sea visible a diferentes funciones con una variable global.
- `global nombre_variable`
- En general, se debe restringir el uso de variables globales a definir constantes (variables de lectura, no escritura).

Ejemplo

```
global GRAVEDAD
GRAVEDAD = 9.81;
```

Visibilidad de las variables

Ejercicio 3

- Agregue un `breakpoint` en la función `int_compuesto`.
- Doble click sobre margen izquierdo del editor.
- Debe aparecer un punto rojo.

```
19
20 ●    Cf = Ci .* (1 + int).^n;
21
22 -    end
```

- Ejecute `prueba_int_compuesto`.
- La ejecución se detiene en el `breakpoint`.
- Analice el contenido del Workspace.
- Oprima la tecla F5 para terminar la ejecución del programa.
- Analice nuevamente el contenido del Workspace. ¿Qué observa?

Comparación entre programa y función

- Ambos se guardan con la extensión `.m` (M-files).
- Una función comienza con una declaración de la función, un programa no.
- Las variables en una función son locales.
- Un programa puede usar variables declaradas en el Workspace con anterioridad.
- Entonces, ¿por qué usar funciones?. Por muchos motivos:
  - Cuando se necesita realizar una misma operación varias veces con diferentes variables.
  - Dividir un problema en partes, «Divide y reinarás».
  - Dividir tareas dentro de un equipo de trabajo.

Importar/exportar datos al/desde el Workspace

`save`

- `save archivo`, guarda en `archivo` todo el Workspace con formato .mat.
- `save archivo variable`, guarda `variable` en `archivo` con formato .mat.
- Por defecto, los archivos se guardan en el directorio de trabajo.
- Si se desea guardar `archivo` en otro directorio:
  `save C:\path\to\file\file.mat`

`load`

- `load archivo`, carga el contenido de `archivo` en el Workspace.
- `load` puede cargar también datos de archivos en texto plano (ASCII), sin importar la extensión (.txt, .csv), siempre y cuando los datos estén ordenados en forma rectangular (matriz).

`delete`

- `delete archivo.mat`, borra permanentemente `archivo`.

Importar/exportar datos al/desde el Workspace

Ejercicio 4

```
» clear
» load data.txt
» data2 = data .* 2
» save all_data
» save data data
» save data2 data2
» clear
» load data2 ¿Que observa en el Workspace?
» clear
» load data ¿Que observa en el Workspace?
» delete data.mat data2.mat
```

Otras formas de importar datos

- Archivo Excell (XLS):
  - `data = xlsread('data.xls').`
- Archivos .csv:
  - `data = csvread('FILENAME', R, C)).`
- Imágenes:
  - `[A,map] = imread('imagen.jpg').`
- Asistente (*Wizard*):
  - `uiimport`, se ejecuta un asistente para la importación de datos al Workspace.
- Y más... (ver en ayuda "Recommended Methods for Importing Data").

Lectura de archivos .csv

Ejercicio 5

Lea los datos del archivo `data.csv` usando la función `csvread`.

Respuesta

1. `data = csvread('data.csv', 1);`

Ejercicio 6

Lea los datos del archivo `gatito.jpg` usando la función `imread`.

Respuesta

1. `[A,map] = imread('gatito.jpg').`

Lectura de archivos

Se desea cargar en el Workspace datos que se encuentran en un archivo en formato de texto plano (ASCII).

- El archivo se abre con,
  - `fid = fopen('data.csv', permiso)`
  - Donde,
    - `fid`, identificador del archivo.
    - `data.csv`, nombre del archivo.
    - `permiso`, puede ser lectura (`'r'`), escritura (`'w'`) u otros.
- El contenido del archivo se lee con,
  - `A = textscan(fid, format)`, función recomendada.
  - `A = fscanf(fid, format, [c f])`
    - `format`: `%d` entero c/signo, `%E` notación científica, `%f` punto flotante, `%s` cadena de caracteres, `%c` caracter, y más.
    - `[c f]`, los datos están ordenados en `c` columnas por `f` filas.
    - `line = fgets(fid)`, lee línea por línea.
- Cuando se termina de leer el archivo se debe cerrar con,
  - `fclose(fid)`

Ejercicio 7

Lea la cabecera y los datos del archivo `data.v2`. Observe que no es posible usar la función `load` ya que la cabecera del archivo (líneas 1 a 46) no posee información en formato rectangular.

Respuesta

```
1 fileID = fopen('data.v2', 'r');
2 header = cell(46, 1);
3 for i =1:46
4  newline = fgets(fileID);
5  header{i, 1} = newline;
6 end
7 samples = 12000;
8 acc = fscanf(fileID, '%f', [8 samples/8])';
9 fclose(fileID);
```

Otras formas de exportar datos

- Archivo Excell (XLS):
  - `status = xlswrite(filename,A,sheet).`
  - `A`, matriz 2D, números o caracteres.
  - Requiere tener instalado Microsoft Excell.
- Imágenes en .pdf, .jpg, .png :
  - `saveas(fig,filename)),`

Ejercicio 8

```
1 data = randn(10)
2 status = xlswrite('midata.xls',data,1)
3 t = 0:0.1:pi;
4 f1=figure;
5 plot(t, cos(t))
6 saveas(f1, 'mi_coseno', 'pdf');
```

Escritura de datos en un archivo

Se desea guardar en un archivo datos que están en el Workspace como texto plano (ASCII).

- El archivo se graba con,
  - `fprintf(fid, formato, data);`.
  - `fid`, identificador del archivo.
  - `formato`. Ver `help fprintf`.

Ejercicio 9

```
1 data = randn(10,4) - 1;
2 fid = fopen('rmse.txt','w');
3 % data se guarda en una matriz de 10x4
4 fprintf(fid,'%5.10f %5.10f %+5.4E %+5.4E\n',data);
5 fclose(fid);
```

Función `sprintf()`

Se utiliza para crear cadena de caracteres a partir de diferentes tipos de datos.

Ejercicio 10

```
1 date_str = date;  % string con dd-Mmm-yyyy
2 hora = clock;  % vector con [año, mes, dia, hora, min, seg]
3 nombre_f = sprintf('archivo_%s_%02d_%02d_%02.f.txt', date_str, hora(4), hora(5), hora(6));
4 fileID = fopen(nombre_f, 'w');
5 fprintf(fileID, 'Este archivo se llama %s \n', nombre_f)
6 for i =1:10
7 fprintf(fileID, 'Esta es la línea número %d \n', i)
8 end
9 fclose(fileID);
```

Función `sscanf()`

Se utiliza para extraer información de una cadena de caracteres a partir de diferentes tipos de datos.

Ejercicio 11

```
1  fileID = fopen('data.v2', 'r');
2  header = cell(46, 1);
3  for i =1:46
4   newline = fgets(fileID);
5   header{i, 1} = newline;
6  end
7  date = sscanf(header{5}(49:end), '%c');
8  lenght = sscanf(header{12}(40:end-6), '%f',1);
9  dt = sscanf(header{17}(33:38), '%f', 1);
10 samples = sscanf(header{16}(1:6), '%f', 1);
11 fclose(fileID);
```

Funciones aleatorias

- Números aleatorios en intervalo [0,1]: `rand(m,n)`
- Números aleatorios en intervalo [a,b]: `(b-a)*rand(m,n)+a`
- Números aleatorios con distribución normal (gaussiana) con media 0 y desviación estándar 1: `randn(m,n)`
- Números aleatorios con distribución normal (gaussiana) con media `a` y desviación estándar σ: σ `* randn(m,n) + a`
- Números aleatorios enteros `randi(imax,m,n)`
- La función `RandStream` fija ciertos parámetros para la posterior generación de números aleatorios. Es opcional.

Funciones aleatorias

Ejercicio 12

```
1 xn = randn(1000,1);
2 hist(xn)
3 pause
4 xa = rand(1000,1);
5 hist(xa)
```

Funciones estadísticas

- Media o promedio: `mean(A)`
- Mediana: `median(A)`
- Moda: `mode(A)`
- Desviación estándar: `std(A)`

Ejercicio 13

```
1 s = RandStream('mt19937ar','Seed',1);
2 RandStream.setGlobalStream(s);
3 xn = 2 * randn(s, 1000, 5) + 5;
4 m1 = mean(xn)
5 m2 = mean(mean(xn))
6 m3 = median(xn)
7 m4 = mode(xn)
8 s1 = std(xn)
```

Ejercicio 14

Cree la función *bolillero* que reciba la cantidad *n* de bolillas que tiene el programa de una materia y entregue un número aleatorio entre 1 y *n*.