vg101: Introduction to Computer Programming

RC 3

ZHANG Yifei

2019/06/02

Outline

- Lectures
    - Function review
        - Function
        - Subfunction
        - Recursion
    - Plotting
    - Advanced topics
        - Data types
        - Structures

Lectures

Function review

Function

- What is a function?
  `A function is a block of organized, reusable code that is used to perform an action.`
  Unlike a mathematical function, writing a function is actually more similar to writing a procedure of doing some spectfic task.

- How to write a function in Matlab?

    1. create a .m file
    2. think about the input, output and intermediate procedure
    3. write the header of the function on the **first line** of the .m file (the name of the function must be the same as the name of the .m file)

    ```
    function [return_values] = function_name(parameters)
    ```

    return_values are the *output* of the function, while *parameters* are the *input* of the function.

4. fill in the body

- How to call a function?
  If you would like to use a function in another script, you can directly call a function.

  ```
  [outputs] = function_name(inputs)
  ```

  Note that you must be in the same folder with the function when you need to call it.

- Function and script

  - script will interact with the variables stored in the workspace (have side effects), while function will not do so (function variables are local to the function itself)
  - script has no input/output arguments, function has

- Call stack
  `Call stack is a stack data structure that stores information about all the functions called by a computer program. It helps each function successfully returns to the function that calls it.`

  - stack: last in first out data structure (like a pile of plates)
  - function call in call stack (assue function A calls function B)
    1. when function A calls function B, we **copy** all the values of the input parameters into B and push B into the stack
    2. perform corresponding operations in function B, compute the return values
    3. when we reach **return** in B or we have already arrived at the end of function B, function B will be popped from the stack, the return value will be received by function A
  - Notices
    1. In the whole process, function B is have no access to any variables of function A
    2. The operation of input parameters of B will not change the corresponding variables in A, since they are a copy of them
    3. Only the return value of B can be received by A

Subfunction

- A function written below the main function (the function with the same name as the filename)
- Only can be called in the current function

Recursion and Iteration

- Iteration: a situation in which some statements are performed repeatedly in a loop

- Recursion: a situation where a function calls itself repeatedly until some base condition is reached.
- How to write recursion algorithms:
    1. Knowing how to break up a big, complex problem into several similar, small, simple pieces.
    2. Knowing when to stop this breaking up process.

Factorial

- Iterative

```
function result=factorial_iteration(n)
  result=1;
  for i = 1:n
      result = result * i;
  end
  return;
end
```

- Recursive

```
function result=factorial_recursion(n)
  if n==1
      result = 1;
      return;
  else
      result = n * factorial_recursion(n-1);
      return;
  end
end
```

Reverse an array

Given a vector or array like [1,2,3,4,5] reverse it.

- Iterative

```
function rst = reverse_iteration(A)
  rst=[];
  for i = 1:length(A)
      rst=[A(i) rst];
  end
  return;
end
```

- Recursive

```
function rst = reverse_recursion(A)
  if length(A)==1
      rst = A;
      return;
  else
      rev = reverse_recursion(A(2:end));
      rst = [rev A(1)];
      return;
  end
end
```

Find smallest element (bisection)

Given a non-empty vector A, return its smallest element.

- Iterative

```
function smallest = find_smallest_iteration(A)
  smallest = A(1);
  for i = 2:length(A)
      if A(i)<smallest
          smallest=A(i);
      end
  end
  return;
end
```

- Recursive

  Algorithm 1: the smallest element is either the first or the smallest of the rest elements in an array.

  Algorithm 2: the smallest element is either the smallest in the first half, or the smallest in the second half.

```
function smallest = find_smallest_recursion(A)
```

```
  if length(A) == 1
      smallest = A(1);
      return;
  else
      left = A(1:floor(length(A)/2));
      right = A(floor(length(A)/2)+1:end);
      left_smallest = find_smallest_recursion(left);
      right_smallest = find_smallest_recursion(right);
      if left_smallest < right_smallest
          smallest = left_smallest;
      else
          smallest = right_smallest;
      end
  end
end
```

Eight queens problem

You are to place 8 chess queens on an 8 × 8 chessboard. These queens refuse to share the same row, the same column or the same diagonal. Find the number of possibilities to place them.

Solution 1:

```
function count=eight_queen(A,row)
  count=0;
  for col=1:8
      col_availability=~(sum(A(:,col)));
      B=zeros(8);
      for a=1:8
          for b=1:8
              if (a-b==row-col)||(a+b==row+col)
                  B(a,b)=1;
              end
          end
      end
      diag_availability=~(sum(A(B==1),'all'));
      if col_availability && diag_availability
          A(row,col)=1;
          if row==8
              count=count+1;
              disp(A);
          else
              count=count+eight_queen(A,row+1);
          end
          A(row,col)=0;
      end
  end
end
```

You can call `eight_queen(zeros(8),1)` to receive the answer.
Solution 2:

```
function count = queen(pre)
  row = size(pre,2) + 1;
  count = 0;
  for col = 1 : 8
      if row == 1
          count = count + queen([pre [row; col]]);
      elseif isempty(find(pre(2,:)==col)) && isempty(find(sum(pre)==row+col)) && isempty(find(pre(1,:)-pre(2,:)==row-col))
          if row == 8
              count = count + 1;
          else
              count = count + queen([pre [row; col]]);
          end
      end
  end
end
```

You can call `queen([])` to receive the answer.

Plotting

Matlab has a whole catalog including all the related information of figures and images named **Graphics** (under Documentation -> MATLAB -> Graphics).

Basic

- figure: create a figure window
- axes: create, and specify the range of axes in the current figure window
- plot: create a new plot, will cover the prev plot in the same figure window

Formatting and Annotation

Titles and labels

- title: add title on the top of a figure (LaTeX syntex supported)
- xlabel/ylabel: set axis labels (LaTeX syntex supported)
- legend: add label (curve name) for different plots in a figure window (LaTeX syntex supported)

Annotations

- text: `text(x,y,txt)` Add text beside a certain data point (x, y)
- line: `line([x0 x1],[y0 ,y1])` Draw a line from (x0, y0) to (x1, y1)
- rectangle: `rectangle('Position',[left_down_x0 left_down_y0 width height])`

  Add in current plot, so no need to use *hold on*

Others

- subplot: `subplot(m, n, p)`
- clf: delete all the graphics objects on the current figure
- hold on/off: retain (or not) the current plot when adding new plots

Advanced topics

Data type

- What is data type? Why we need it?
  - Each variable has a specific data type
  - It tells the interpreter how to allocate memory and how to load and interpret some bits at certain address in memory
  - It defines the operations a certain variable can have
- Different data types in Matlab
  - int: int8, int16, int32, int64

    The most significant bit is the sign bit.
  - uint: uint8, uint16, uint32, uint64
  - single(32 bit), double(64 bit)
  - logical
  - char (single quotation mark), string(double quotation mark)
    - provide storage for text data, but string array treats each phrase as a unit, whereas a char array treats each character as a unit
    - their ways for storage data is likely to be different

      eg: Try this in command window

```
chr='abcd';
chr(1)
whos chr
str="abcd";
str(1)
whos str
```

        - string specfic functions are both suitable for *string* and *char array* (like str2num, strcmp, strfind, strrep), but the data type of the return value may be affected by your choice
    - cell array
    - structure
    - function handle
    - classes
- Type Conversion
  There are many ways to convert one data type into another in MATLAB.
  eg. convert string into double

```
str = "123.45";
db = str2double(str); % similar to int2str, mat2str, num2str, str2num

db = cast(str,'double')

db = double(str);
```

Structures

- What is a structure?
  A structure is a data type that groups related data using data containers called fields
- Initialize a structure
    - specify the name of each field and corresponding values
    - all elements in a typical structure array should have same fields
    - field names should be *char array* or *string*

```
students(1) = struct('name',"Jane",'id',"01",'hw',[80,90,80],'mid',100,'final',90);
students(2) = struct('name',"Simon",'id',"02",'hw',[95,85,100],'mid',80,'final',100);
students(3) = struct('name',"Alice",'id',"03",'hw','None','mid',95,'final',95);
% students(4) = struct('name','AAA','id','04');  % will cause an error
% However, the following is correct
students(4).name = "Bob";  % all other fields will be left as empty []
students(4).id = "04";
students(4).hw = [10 20 30];
```

- Access and modify a field in structure

```
% access/modify a field
students(2).name="XXX";
name_list = [students.name]; % obtain an array of all the names
```