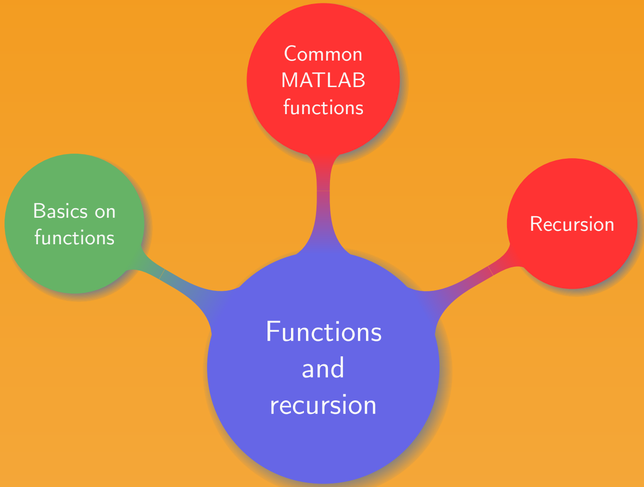
Introduction to Computer and Programming

3. Functions and recursion

Manuel – Summer 2019

Chapter organisation



From script to function

Script:

- Sequence of MATLAB statements
- No input/output arguments
- Operates on data on the workspace

From script to function

Script:

- Sequence of MATLAB statements
- No input/output arguments
- Operates on data on the workspace

Function:

- Sequence of MATLAB statements
- Accepts input/output arguments
- Variable are not created on the workspace

Functions in MATLAB

Basics on MATLAB functions:

- Function saved in a .m file
- The .m file must be in the “path”
- The function name must be the same as the filename
- Prototype: `function [out1,out2,...] = Myfct(in1,inp2,...)`
- Functions can be called from an .m file or from the workspace

Functions in MATLAB

Basics on MATLAB functions:

- Function saved in a .m file
- The .m file must be in the "path"
- The function name must be the same as the filename
- Prototype: `function [out1,out2,...] = Myfct(in1,inp2,...)`
- Functions can be called from an .m file or from the workspace

Script

```
r=1.496*10^11; c=4.379*10^9;
G=6.674*10^-11;
T=365*24*3600;
V=4*pi/3*(c/(2*pi))^3;
M=4*pi^2*r^3/(G*T^2);
M/V
```

Function

density.m

```
function d=density(r,c,T)
  G=6.674*10^-11;
  V=4*pi/3*(c/(2*pi))^3;
  M=4*pi^2*r^3/(G*T^2);
  d=M/V;
```

Sub-functions

A .m file can contain:

- A main function: has the same name as the filename
- Sub-functions: only accessible by functions from the **same** file

Exercise.
For a vector, write a function returning the mean and the standard deviation. Calculate the mean in a sub-function

Sub-functions

A .m file can contain:

- A main function: has the same name as the filename
- Sub-functions: only accessible by functions from the **same** file

Exercise.
For a vector, write a function returning the mean and the standard deviation. Calculate the mean in a sub-function

stat.m

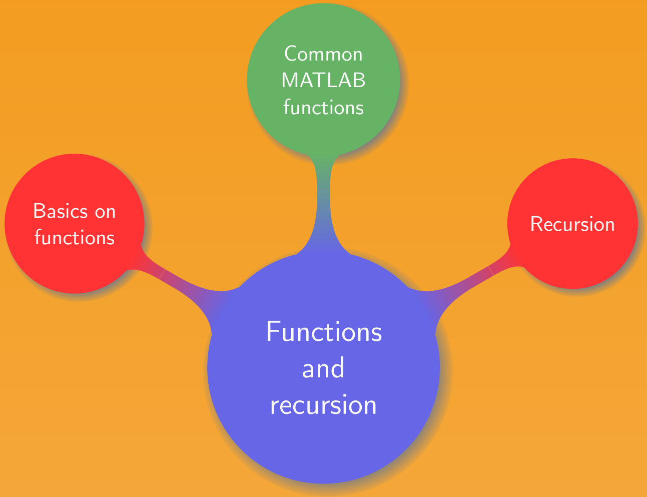
```
function [mean,stdev] = stat(x)
    n = length(x);
    mean = avg(x,n);
    stdev = sqrt(sum((x-mean).^2)/n);

function  mean = avg(x,n)
    mean = sum(x)/n;
```

Functions and sub-functions

In the previous example:

- How to save both the variable mean and stdev?
- How many Input have the `avg` and `stat` functions?
- Is the function `avg` accessible from the workspace, why?
- If `mean` is changed into `m` in the first function does it need to be changed in the second function, why?

Chapter organisation



Mathematical functions

Basic math calculations:

- Defining a function: `f=@(x) x^2-1`
- Integral: `syms z; int(z^2+1), int(z^2+1,0,1)`
- Differentiation: `syms t; diff(sin(t^2))`
- Limit: `limit(sin(t)/t,0)`
- Finding a root of a continuous function: `fzero(f,0.5)`
- Square root: `sqrt(9)`
- Nth root: `nthroot(4, 3)`

Useful functions

The `save` and `load` functions:

- Save variables: `save('file','var1','var2',...,'format')`
- Load variables: `load('file','format')`

Useful functions

The `save` and `load` functions:

- Save variables: `save('file','var1','var2',...,'format')`
- Load variables: `load('file','format')`

Random number generation:

- An $n \times m$ matrix of random numbers: `rand(n,m)`
- Random numbers following a specific distribution `dist`: `random('dist',parameters)`
- Random numbers initialized with a specific seed: `rand('state',datenum(clock))`
- A random permutation: `randperm(n)`

The `sprintf` function

Writing formatted data into a string:

- Command: `sprintf('string', variable1, variable2,...)`
- `'string'`: text composed of
  - Words, spaces, numbers
  - “% flags”, replaced by the value of variables, e.g. `'%g'`
  - Special characters, e.g `'\n\t'`

The `sprintf` function

Writing formatted data into a string:

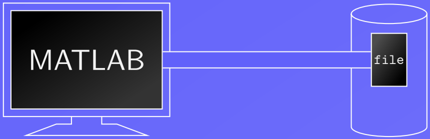
- Command: `sprintf('string', variable1, variable2,...)`
- `'string'`: text composed of
  - Words, spaces, numbers
  - "% flags", replaced by the value of variables, e.g. `'%g'`
  - Special characters, e.g `'\n\t'`

Example.

```
a=pi; b=sprintf('%g',pi)
sprintf('%d',round(pi))
sprintf('%s','pi')
a=[1 2 3;2 5 6;3 7 8];
text=sprintf('size: %d by %d', size(a))
```

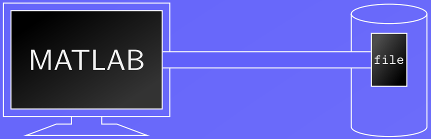
File input/output

Open a stream between MATLAB and a file



File input/output

Open a stream between MATLAB and a file



```
fd=fopen('file.txt', 'permission')
fclose(fd)
```

Different permissions to access a file:

- Read only: `r`
- Write in a new file: `w`
- Append to a file: `a`
- Read and write: `r+`
- Read and overwrite: `w+`
- Read and append: `a+`

The `fprintf` and `fscanf` functions

Accessing a file:

- Write: `fprintf(fd, 'string', 'variables')`
- Read:
    - Following a known format: `fscanf(fd, 'format')`
        - Convert values into the specified format
        - Return an array containing the read elements
    - A whole line: `fgetl(fd)`

The `fprintf` and `fscanf` functions

Accessing a file:

- Write: `fprintf(fd, 'string', 'variables')`
- Read:
  - Following a known format: `fscanf(fd, 'format')`
    - Convert values into the specified format
    - Return an array containing the read elements
  - A whole line: `fgetl(fd)`

Any opened stream must be closed

File reading and writing

Exercise.
Given a text file where each line is composed of three fields, first-name, name and email, write a MATLAB function generating a text file where (i) the order of the lines is random and (ii) each line is composed of the same fields in the following order: name, first-name, and email.

File reading and writing

sortnames.m

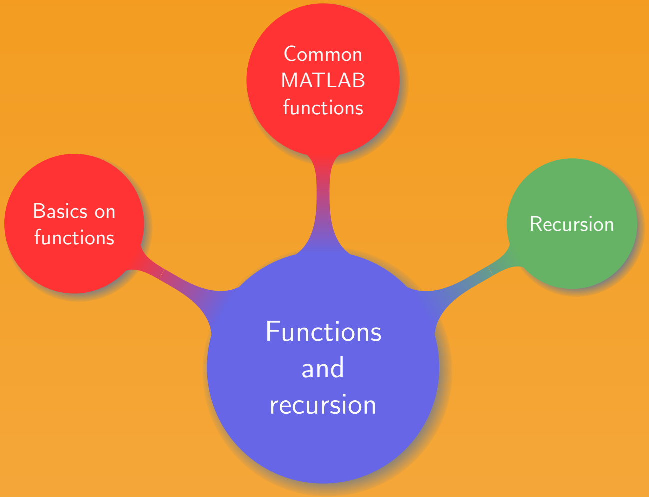
```
function sortnames(finput, foutput)
  fd1=fopen(finput,'r');
  i=1;
  line=fgetl(fd1);
  while line ~= -1
    a=find(isspace(line),2);
    info{i}=sprintf('%s %s %s\n', line(a(1)+1:a(2)-1), ...
      line(1:a(1)-1), line(a(2)+1:end));
    i=i+1; line=fgetl(fd1);
  end
  fclose(fd1);

  fd2=fopen(foutput,'w');
  for j=randperm(i-1)
    fprintf(fd2,info{j});
  end
  fclose(fd2);
```

File reading and writing

Understanding the code:

- How is the code indented?
- How to check the last line was reached, why?
- How to access the different fields?
- How to perform a random permutation?
- Each time a file is opened it **must** be ____

Chapter organisation



What is recursion?

Droste
HAARLEM-HOLLAND
cacao
Netto 250 g
What is recursion?

Recursive acronyms

Famous acronyms:

- GNU: GNU's Not Unix
- WINE: WINE Is Not an Emulator
- PHP: PHP Hypertext Preprocessor
- LAME: LAME Ain't an MP3 Encoder

Recursion in computer science

Basic idea behind recursion:

- General: given a process P and some data D, describe P using itself together with a simplified version D' of D
- Computer science: inside a function `P(D)`, call the function `P(D')`

Recursion in computer science

Basic idea behind recursion:

- General: given a process P and some data D, describe P using itself together with a simplified version D' of D
- Computer science: inside a function `P(D)`, call the function `P(D')`

Example.
A child couldn't sleep, so her mother told her a story about a little frog, who couldn't sleep, so the frog's mother told her a story about a little bear, who couldn't sleep, so the bear's mother told her a story about a little weasel...who fell asleep. ...and the little bear fell asleep; ...and the little frog fell asleep; ...and the child fell asleep.

A short recursive story

For the sake of simplicity we work with integers and map the child to 3, the frog to 2, the bear to 1, and the weasel to 0.

Algorithm. (*Bedtime story*)

Input : An integer n representing an animal or a child
Output: The child and all the animals asleep

```
Function Read($n$):
    if $n = 0$ then sleep(n);
    else $i \leftarrow n - 1$; Read($i$); sleep($n$);
end
```

Exercise.
Draw a simple diagram showing how recursion is applied

Numbers in words

For an automated information service a telephone company needs the digits of phone numbers to be read digit by digit. Therefore you are asked to rewrite a sequence of digits into words, with a space between each word; no space at the beginning and at the end.

Numbers in words

Algorithm. (*Numbers in words*)

Input : A large integer n
Output : n, digit by digit, using words

```
Function PrintDigit(n):
    case n do
        0: print('zero'); 1: print('one'); 2: print('two'); 3: print('three');
        4: print('four'); 5: print('five'); 6: print('six'); 7: print('seven');
        8: print('eight'); 9: print('nine'); else: error('not a digit');
    end case
end
Function PrintDigits(n):
    if $n < 10$ then
        PrintDigit (n)
    else
        PrintDigits ($n$ div 10);
        print(' '); PrintDigit ($n$ mod 10)
    end if
end
```

Recursion vs. iteration

When to prefer recursion over iteration:

- A recursive algorithm is more obvious than an iterative one
- Depends on the language

MATLAB, C, and C++:

- Deal best with iterative
- Can run recursive algorithm without nay problem
- Prefer iterative over recursive when facing two equivalent solutions

Recursion vs. iteration

When to prefer recursion over iteration:

- A recursive algorithm is more obvious than an iterative one
- Depends on the language

MATLAB, C, and C++:

- Deal best with iterative
- Can run recursive algorithm without nay problem
- Prefer iterative over recursive when facing two equivalent solutions

When using recursion pay attention to the memory usage

Key points

- Why should functions be preferred over scripts?
- How to perform mathematical calculations in MATLAB?
- How to save the state of the workspace?
- What is recursion?
- When to use recursion?

Thank you!

References I

3.17 https://upload.wikimedia.org/wikipedia/commons/0/0f/Droste_cacao_100gr_blikje%2C_foto_02.JPG

3.18 https://upload.wikimedia.org/wikipedia/en/2/22/Heckert_GNU_white.svg