Intro. to Computer Programming Midterm 1

CHEN Xiwen

UM-SJTU Joint Institute

June 12, 2019

Table of contents

Functions

- Defining Functions
- Useful Functions

File I/O

- Workspace
- File Reading and Writing

Recursion

- Methodology
- Examples

Remarks

- Functions
  - Defining Functions
  - Useful Functions
- File I/O
  - Workspace
  - File Reading and Writing
- Recursion
  - Methodology
  - Examples
- Remarks

Defining Functions

- **Via a file.** `myFunct.m` possibly with sub-functions:

```
    function [out1, out2, ...] = myFunct(in1, in2, ...)
```

- **Via function handle.**

```
    f = @(arg1, arg2, ...) expression(arg1, arg2, ...);
    result = f(in1, in2, ...);
```

Functions

- Defining Functions
- Useful Functions

File I/O

- Workspace
- File Reading and Writing

Recursion

- Methodology
- Examples

Remarks

Useful Functions

- **Random integers.**

```
randi([minVal, maxVal], rows, cols);
```

- **Formatted output.**

```
fprintf(formatSpec, values);
```

Commonly used formats: `%d`, `%f`, `\n`, `...`

- Functions
    - Defining Functions
    - Useful Functions

- File I/O
    - Workspace
    - File Reading and Writing

- Recursion
    - Methodology
    - Examples

- Remarks

Workspace

- Command.

```
   save(filename, var1, var2, ..., formatSpec);
   load(filename, formatSpec);
```

- What are the optional formats?
- What are the types of the stored data? What will happen if you reload the data?

Functions

- Defining Functions
- Useful Functions

File I/O

- Workspace
- File Reading and Writing

Recursion

- Methodology
- Examples

Remarks

File Reading and Writing

- Open and close a file.

```
   fid = fopen(filename, permission);
   fclose(fid);
```

- Reading from a file.

```
   fscanf(fid, formatSpec, variables);
   fgetl(fid);
   fread(fid, sizeA, precision);
```

- Write to a file.

```
   fprintf(fid, formatSpec, variables);
   fwrite(fid, A, precision)
```

File Reading and Writing

Example

- Formatted I/O.

```
d = date();
fid = fopen(’date.txt’, ’r’);
fprintf(fid, ’Today is %s.\n’, d);
fclose(fid);
```

- Binary write/read.

```
A = magic(4);
fid = fopen(’magic.txt’, ’w’);
fwrite(fid, A, ’int64’);
fclose(fid);
```

Q: Once you have written these data to the files, how can you read the original data back from these output files?

Functions

- Defining Functions
- Useful Functions

File I/O

- Workspace
- File Reading and Writing

Recursion

- Methodology
- Examples

Remarks

Methodology

- Required to design a function f to solve a problem of size D_n.
- Assume we know how to solve of problem of size smaller than or equal to D_{n-1}. (How can we solve the problem of size D_n with solutions of problem of sizes $D_1, \ldots, D_{n-1}$?)
- Then to solve the current problem of size D_n,
    1. Call $f(D_{n-1})$ and possibly $f(D_{n-2}), f(D_{n-3}), \ldots$
    2. Solve problem of size D_n with the solutions to smaller-sized problems.

Methodology

In terms of programming,

- Identify base case: the smallest problem that we can solve without question.
- Design algorithm to solve the problem of size D_n with previous solutions. Get the solutions we need by calling the function inside itself.

Functions

- Defining Functions
- Useful Functions

File I/O

- Workspace
- File Reading and Writing

Recursion

- Methodology
- Examples

Remarks

Generate Permutations

Algorithm 1 Generate Permutations

```
Require: a set of characters $S$
Ensure: a set containing all permutations of the characters
if $S$ is empty or contains only one character then
    return $S$;
end if
for all character $c \in S$ do
    call Generate Permutations with the remaining characters;
    concatenate $c$ with all returned permutations and set as $P_c$;
end for
merge all $P_c$s into a single set $P$;
return $P$;
```

Depth-First Search

Algorithm 2 Depth-First Search (DFS)

```
Require: graph $G$, with some nodes connected, a source node $s$
Ensure: visit all the nodes in the graph
visit $s$;
update $G$;
for all node $v$ that can be reached from $s$ do
    if node $v$ has not been visited then
        call DFS with the current graph $G$ and source node $v$;
    end if
end for
```

Factorization

Q: What is the problem in the following pseudocode?

Algorithm 3 Factorize Integers

```
Require: a positive integer $n$
Ensure: a set of primes $P$ with $n = p_1 \times p_2 \times \cdots \times p_k$
  if $n$ is a prime or 1 then
    return $\{n\}$;
  end if
  $P \leftarrow \{\}$;
  for all $k \leftarrow 1, \ldots, \lfloor \sqrt{n} \rfloor$ do
    if $n$ is divisible by $k$ then
      add $k$ into $P$;
      call Factorization with input $n/k$;
      merge the output from the previous call with $P$;
      return $P$;
    end if
  end for
```

Factorization

Algorithm 4 Factorize Integers

Require: a positive integer n

Ensure: a set of primes P with $n = p_1 \times p_2 \times \cdots \times p_k$

```
if $n$ is a prime or 1 then
    return $\{n\}$;
end if
$P \leftarrow \{\}$;
for all $k \leftarrow 2, \ldots, \lfloor \sqrt{n} \rfloor$ do
    if $n$ is divisible by $k$ then
        add $k$ into $P$;
        call Factorization with input $n/k$;
        merge the output from the previous call with $P$;
        return $P$;
    end if
end for
```

Remarks

- Slides with *relevant* notes are allowed in Part B. But do not write the whole project on your notes. :)
- Think about the questions on the last slide of each chapter.
- Do not expect to complete all the questions.
- Try to be more familiar with coding.
- If you encounter something that is unfamiliar to you, search in documentation.
- *Read through the exam paper before you start.*

Good luck for your Midterm 1!