Algoritma & Pemrograman Saintifik

Gatot F. Hertono, Ph.D

Departemen Matematika

SCMA601401

Definition

Functions are "self contained" modules of code that accomplish a specific task.

- Functions usually "take in" data, process it, and "return" a result.
- Once a function is written, it can be used over and over and over again.
- Functions can be "called" from the inside of other functions.

$$f(x) = x^2 - 10x + 5$$

```
function NameOfFunction ( list_of_parameters )
        ....
end function
```

Main.m

```
a=input('masukkan suatu integer: ')
Sol1 = MyFunc(a)
Sol2 = MyFunc(a)+MyFunc(2*a)
```

"Take in data"

```
function Answer = MyFunc(x)
Answer = (x^2)-(10*x)+5;
```

"return a result"

"Process"

MyFunc.m

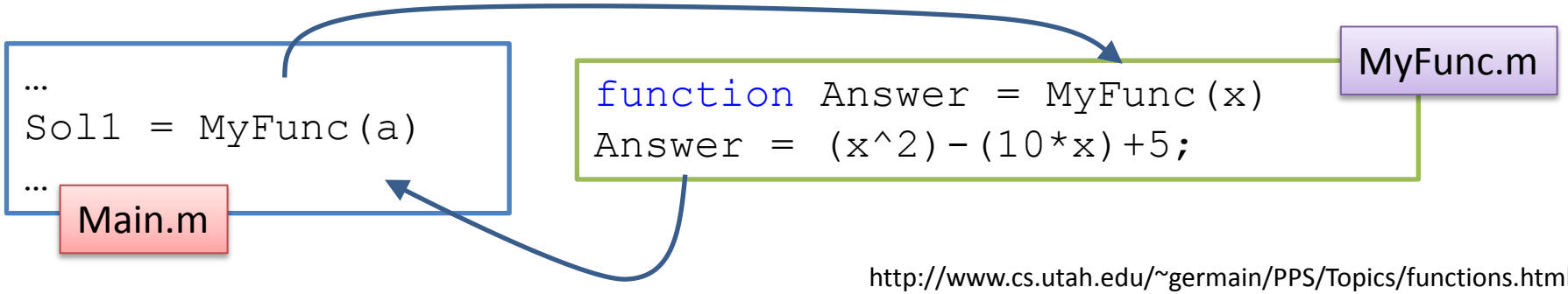
How does a function work?

Functions "Encapsulate" a task (they combine many instructions into a single line of code). Most programming languages provide many **built in functions**(e.g. sin(), cos(), sqrt() etc.)

When a function is "called" the program "leaves" the current section of code and begins to execute the first line inside the function.

Thus the function "flow of control" is:

1. The program comes to a line of code containing a "function call".
2. The program enters the function (starts at the first line in the function code).
3. **All instructions** inside of the function are executed from top to bottom.
4. The program leaves the function **and goes back to where it started from.**
5. Any data computed and **RETURNED** by the function is used in place of the function in the original line of code.



http://www.cs.utah.edu/~germain/PPS/Topics/functions.html

Why do we write functions?

1. They allow us to conceive of our program as a bunch of sub-steps. (Each sub-step can be its own function. When any program seems too hard, just break the overall program into sub-steps!)

2. They allow us to reuse code instead of rewriting it.

3. Functions allow us to keep our variable namespace clean (local variables only "live" as long as the function does). In other words, function_1 can use a variable called *i*, and function_2 can also use a variable called i and there is no confusion. Each variable *i* only exists when the computer is executing the given function.

4. Functions allow us to test small parts of our program in isolation from the rest. This is especially true in interpreted languages, such as Matlab, but can be useful in C, Java, ActionScript, etc.

http://www.cs.utah.edu/~germain/PPS/Topics/functions.html

Examples

Staff	Week 1	Week 2	Week 3	Week 4
Staff 1	55	40	43	35
Staff 2	60	40	40	40
Staff 3	40	40	50	55
Staff 4	55	55	40	35
Staff 5	45	40	40	40

Standard Rate=Rp.1,000,000,-/jam

Overtime bonus (> 40 jam) = 1.5*Standard Rate/jam

```
DataStaf = [55 40 43 35; 60 40 40 40; 40 40 50 55;
            55 55 40 35; 45 40 40 40];
NumbOfWeek=4;
NumbOfStaff=size(DataStaf,1);
TotalWeekly=zeros(NumbOfWeek,1);,
TotalMonthly=0;

for i=1:NumbOfWeek
   for j=1:NumbOfStaff
       Salary(i,j) = WeekHour(DataStaf(j,i));
       TotalWeekly(i)=TotalWeekly(i)+Salary(i,j);
   end;
   TotalMonthly = TotalMonthly + TotalWeekly(i);
end;
Salary
TotalWeekly
TotalMonthly
```

Payroll.m

```
function WeekSal = WeekHour(jam)
Rate = 1000000;

if (jam <= 40)
  WeekSal = jam*Rate;
else
  WeekSal =
 (40*Rate)+((jam-40)*1.5*Rate);
end;
```

WeekHour.m

Formal vs. Actual Parameters

Payroll.m

Actual parameters

```
for i=1:NumbOfWeek
    for j=1:NumbOfStaff
        Salary(i,j) = WeekHour(DataStaf(j,i));
        TotalWeekly(i)=TotalWeekly(i)+Salary(i,j);
    end;
    TotalMonthly = TotalMonthly + TotalWeekly(i);
end;
```

```
function WeekSal = WeekHour(jam)
    …
    …
end function
```

Formal parameters

WeekHour.m

Notes:

- Actual parameters could be an expression, constants, or variables, BUT formal parameters have to be variables

Nested Functions?

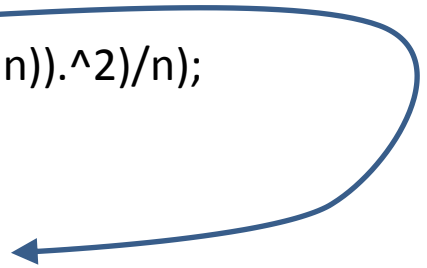
```
% Here is pseudocode of the correct layout of two functions
        function1()
          code;
          code;
          code;
        end;

        function2()
          code;
          code;
          code;
        end;
```

```
% Here is pseudocode of the INCORRECT layout of two functions

function1()
  code;
  code;
  code;
        % forgot to end the function properly with
function2()
  code;
  code;
  code;
end;
end % accidentally ended the first function AFTER the second!
```

Subfunctions are not visible outside the file where they are defined.

Examples

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


        %------------------------
        function mean = avg(x,n)
          %AVG subfunction
          mean = sum(x)/n;
        end;
```



It is saved in one .M file

A subfunction that is visible to the other functions in the same file is created by defining a new function with the function keyword after the body of the preceding function or subfunction.

Recursive Functions

```
n1 = input('Masukkan sebuah integer pertama: ');
n2 = input('Masukkan sebuah integer kedua: ');

Solusi1 = Fakt(n1);
Solusi2 = Fakt(n2);
sprintf('Nilai %d faktorial = %d', n1, Solusi1)
sprintf('Nilai %d faktorial = %d', n2, Solusi2)
```

```
function FN = Fakt(n)
if (n==0)
    FN = 1;
else
    FN = Fakt(n-1)*n;
end;
```