

Function (1A)

Copyright (c) 2010 – 2012 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Task: Finding Partial Sums (1)

$$S_n = \sum_{k=1}^n a_k$$

$$a_k = k$$

$$S_1 = \sum_{k=1}^1 k = 1$$

```
printf("S1 = %d \n", S1);
```

$$S_2 = \sum_{k=1}^2 k = 1 + 2$$

```
printf("S2 = %d \n", S2);
```

$$S_3 = \sum_{k=1}^3 k = 1 + 2 + 3$$

```
printf("S3 = %d \n", S3);
```

Task: Finding Partial Sums (2)

$$S_1 = \sum_{k=1}^1 k = 1$$

$$S_2 = \sum_{k=1}^2 k = 1 + 2$$

$$S_3 = \sum_{k=1}^3 k = 1 + 2 + 3$$

```
S1 = 0;  
for k=1:1      S1 += k;      endfor
```

```
printf("S1 = %d \n", S1);
```

```
S2 = 0;  
for k=1:2      S2 += k;      endfor
```

```
printf("S2 = %d \n", S2);
```

```
S3 = 0;  
for k=1:3      S3 += k;      endfor
```

```
printf("S3 = %d \n", S3);
```

Task: Finding Partial Sums (3)

```
1;  
n ↲;  
S = 0;  
for k=1:n S += k; endfor  
↳ S;
```

```
printf("S1 = %d \n", S1);
```

```
2;  
n ↲;  
S = 0;  
for k=1:n S += k; endfor  
↳ S;
```

```
printf("S2 = %d \n", S2);
```

```
3;  
n ↲;  
S = 0;  
for k=1:n S += k; endfor  
↳ S;
```

```
printf("S3 = %d \n", S3);
```

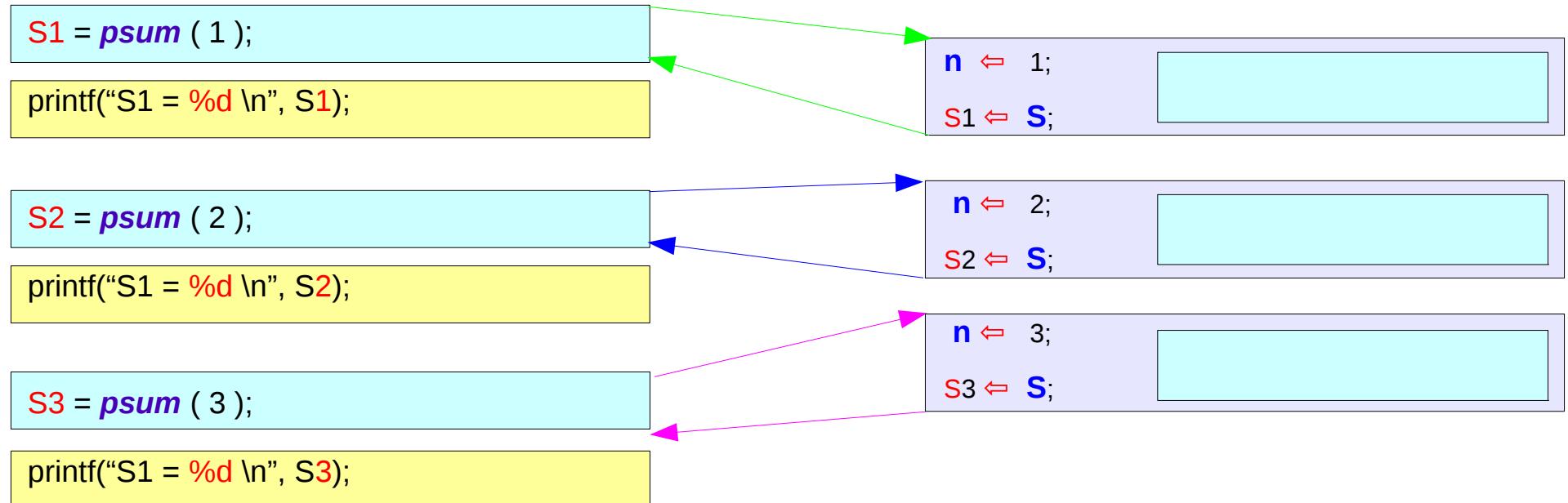
variable name
to be returned = function
name (argument
list)

```
function S = psum ( n )
```

```
S = 0;  
for k= 1:n S += k; endfor
```

```
endfunction
```

Task: Finding Partial Sums (4)



```
function S = psum ( n )
    S = 0;
    for k= 1:n S += k; endfor
endfunction
```

```
↙ █;
n = input_n;
S = 0;
for k= 1:n S += k; endfor
↙ S;
```

Function Files

```
S1 = psum ( 1 );
printf("S1 = %d \n", S1);
S2 = psum ( 2 );
printf("S2 = %d \n", S2);
S3 = psum ( 3 );
printf("S3 = %d \n", S3);
```

Since **psum** identifier is declared,
psum can be **used** here.

function **S** = psum (**n**)

```
S = 0;
for k= 1:n S += k; endfor
```

endfunction

psum.m function file

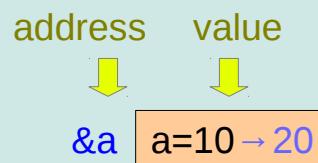
Subfunctions in a function file

src1.c

src2.c

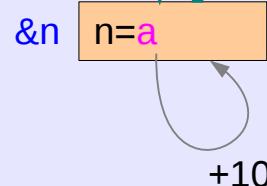
Call by Value

```
a = 10;  
  
printf("a = %d \n", a);  
a = func (a);  
printf("a = %d \n", a);
```



the **value** of **a** is passed through the parameter variable **n** and receives the return value into **a**

```
function n = func (n)  
  
printf("n = %d \n", n);  
n += 10;  
printf("n = %d \n", n);  
  
endfunction
```



n is **local** to the function **func** and exists **while** the function is being **called**
But the return variable is declared as **n**

Return with many variables

```
a = [ 1, 2, 3, 4, 5];  
  
disp(a);  
[amax, aindx] = func ( a );  
printf("amax = %d \n", amax);  
printf("aindx = %d \n", aindx);
```

```
function [max, idx] = func (v)  
idx = 1;  
max = v (idx);  
for i = 2:length (v)  
    if (v (i) > max)  
        max = v (i);  
        idx = i;  
    endif  
endfor  
endfunction
```

[return variable list] = function name (argument list)

Function Calls in Octave (1)

```
int x, y;
```

call by value

x y

S = *func(a, b)*

~~call by reference~~

Function handle: A pointer to a function

`f = @sin;` a function handle `f`
 that refers the function `sin()`

`feval(f, pi/4);`

`f(pi/4);`

Anonymous Functions

$$f(x) = x^2 + x + 1$$

Mathematical Expression

```
f = @(x) x^2 + x + 1;
```

Octave Expression

f = @(arg1, arg2, ...) <function expression>

```
f = @(x, y) x^2 + x*y + y^2;
```

```
f(1, 1);
```

References

- [1] Essential C, Nick Parlante
- [2] Efficient C Programming, Mark A. Weiss
- [3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
- [4] C Language Express, I. K. Chun