

CSE1301
Computer Programming
Lecture 13
Functions (Part 1)

1

Topics

- Functions
- Parameters
- Return values

2

User-Defined Functions

- Create your own functions, similar to `printf()` or `sqrt()`
- Recall a *procedure* in an algorithm - a named collection of instructions
 - InviteToParty
 - RingUp
 - MakeToParty
- A function implements the procedure or function parts of an algorithm.

3

Writing User-defined Functions

- Need to specify:
 - the *name* of the function
 - its *parameters*
 - what it *returns*
 - *block* of statements to be carried out when the function is called
- The block of statements is called the “*function body*”

4

Example: hello1.c

Prints a simple greeting.

```
procedure sayHello
{
  output "Hello World!"
}

Main Program
{
  do procedure sayHello
}
}
```

5

Example: hello1.c

```
#include <stdio.h>

/*
 * Print a simple greeting.
 */

void sayHello ( void )
{
  printf("Hello World!\n");
}

/*
 * Call a function which
 * prints a simple greeting.
 */

int main(void)
{
  sayHello();
  return 0;
}

Prints a simple greeting.

procedure sayHello
{
  output "Hello World!"
}

Main Program
{
  do procedure sayHello
}
}
```

6

Example: hello1.c

```
#include <stdio.h>

/*
 * Print a simple greeting.
 */
void sayHello ( void )
{
    printf("Hello World!\n");
}

/*
 * Call a function which
 * prints a simple greeting.
 */
int main(void)
{
    sayHello();
    return 0;
}
```

Function definition (points to the `void sayHello (void)` block)

Function call (points to the `sayHello();` line in `main`)

7

Example: hello1.c

```
#include <stdio.h>

/*
 * Print a simple greeting.
 */
void sayHello ( void )
{
    printf("Hello World!\n");
}

/*
 * Call a function which
 * prints a simple greeting.
 */
int main(void)
{
    sayHello();
    return 0;
}
```

Function name (points to `sayHello` in the declaration)

Function body (points to the curly braces containing the `printf` statement)

8

Example: hello1.c

```
#include <stdio.h>

/*
 * Print a simple greeting.
 */
void sayHello ( void )
{
    printf("Hello World!\n");
}

/*
 * Call a function which
 * prints a simple greeting.
 */
int main(void)
{
    sayHello();
    return 0;
}
```

Return type (points to `void` in the declaration)

Formal Parameter List (points to `sayHello (void)`)

9

Parameters

- Information passed to a function
- “Formal” parameters are local variables declared in the function declaration.
- “Actual” parameters are values passed to the function when it is called.

10

Example: badsort.c

```
/* Print two numbers in order. */
void badSort ( int a, int b )
{
    int temp;
    if ( a > b )
    {
        printf("%d %d\n", b, a);
    }
    else
    {
        printf("%d %d\n", a, b);
    }
}
```

Parameters (aka Arguments) (points to `int a, int b`)

11

Example: badsort.c

```
/* Print two numbers in order. */
void badSort ( int a, int b )
{
    int temp;
    if ( a > b )
    {
        printf("%d %d\n", b, a);
    }
    else
    {
        printf("%d %d\n", a, b);
    }
}
```

12

Example: badsort.c

```

/* Print two numbers in order
*/
void badSort ( int a, int b )
{
    int temp;

    if ( a > b )
    {
        printf("%d %d\n", b, a);
    }
    else
    {
        printf("%d %d\n", a, b);
    }
}

int main(void)
{
    int x = 3, y = 5;
    badSort ( 10, 9 );
    badSort ( y, x+4 );
    return 0;
}
    
```

Formal parameters (points to 'a' and 'b' in the function signature)

Actual parameters (points to '10, 9' and 'y, x+4' in the function calls)

13

Parameters (cont.)

- Parameters are passed by **copying** the value of the actual parameters to the formal parameters.
- Changes to formal parameters do not affect the value of the actual parameters.

14

Example: badswap.c

```

/* Swap the values of two
variables. */
void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}

int main(void)
{
    int a = 3, b = 5;

    printf("%d %d\n",a,b);
    badSwap ( a, b );
    printf("%d %d\n",a,b);

    return 0;
}
    
```

15

Example: badswap.c

```

/* Swap the values of two
variables. */
void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}

int main(void)
{
    int a = 3, b = 5;

    printf("%d %d\n",a,b);
    badSwap ( a, b );
    printf("%d %d\n",a,b);

    return 0;
}
    
```

Output: 3 5

16

Example: badswap.c

```

/* Swap the values of two
variables. */
void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}

int main(void)
{
    int a = 3, b = 5;

    printf("%d %d\n",a,b);
    badSwap ( a, b );
    printf("%d %d\n",a,b);

    return 0;
}
    
```

Output: 3 5
5 3

17

Example: badswap.c

```

/* Swap the values of two
variables. */
void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}

int main(void)
{
    int a = 3, b = 5;

    printf("%d %d\n",a,b);
    badSwap ( a, b );
    printf("%d %d\n",a,b);

    return 0;
}
    
```

Output: 3 5
5 3
3 5

18

Example: badswap.c

<pre> /* Swap the values of two variables. */ void badSwap (int a, int b) { int temp; temp = a; a = b; b = temp; printf("%d %d\n", a, b); } </pre>	<pre> int main(void) { int a = 3, b = 5; printf("%d %d\n",a,b); badSwap (a, b); printf("%d %d\n",a,b); return 0; } </pre>
--	---

Called function's environment:

a: 5

b: 3

Calling function's environment:

a: 3

b: 5

19

Parameters (cont.)

- If a function does not take parameters, declare its formal argument list **void**.

Declaration:

```

void sayHello ( void )
{
    printf("Hello World!\n");
}
        
```

Function call:

```

sayHello();
        
```

20

Return Values

- Values are returned by copying a value specified after the **return** keyword

21

Example: max.c

Return type

```

/* Returns the larger of two
numbers. */
int max (int a, int b)
{
    int result;

    if (a > b)
    {
        result = a;
    }
    else
    {
        result = b;
    }

    return result;
}
        
```

22

Example: max.c

```

/* Returns the larger of two
numbers. */
int max (int a, int b)
{
    int result;

    if (a > b)
    {
        result = a;
    }
    else
    {
        result = b;
    }

    return result;
}
        
```

For example:
The value of the expression
max(7,5)
is the integer 7.

23

Example: max.c

This style okay.

```

/* Returns the larger of two
numbers. */
int
max (int a, int b)
{
    int result;

    if (a > b)
    {
        result = a;
    }
    else
    {
        result = b;
    }

    return result;
}
        
```

24

Return Values (cont.)

- If a function does not return a value, declare its return type `void`.

Declaration: `void sayHello (void)`
`{`
`printf("Hello World!\n");`
`}`

Function call: `sayHello();`

25

Reading for this lecture

- Forouzan & Gilberg: Chapter 4 (4.2-4.3)
- Deitel & Deitel: Chapter 5 (5.1-5.5)

26