

Day07 A

Young W. Lim

2017-09-21 Thr

- 1 Based on
- 2 Introduction
 - C Functions

"C How to Program", Paul Deitel and Harvey Deitel

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head 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.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

Program Modules in C

- to divide a large program into several smaller program modules
- manageable modules
- modules are functions in C
- a function is invoked by a function call
 - mentions the function by name
 - provides necessary information (arguments)
- information hiding
 - hide detailed information

Function Call

- the function's name (a comma separated list of arguments)
- each argument of a function
 - a constant
 - a variable
 - an expression
- the arguments passed to a function should match in number, type, and order with the parameters in the function definition
- control is transferred from the point of invocation to the called function
 - the statements of the called functions are executed
 - at the end, the control returns to the caller

Function Return

- a called function return control to the caller
- when return no value, control is returned
 - when the function ending right brace is reached
 - or by executing `return` statement
- when the function returns a value
 - by executing `return` expression

- a local variable is known only in a function definition
- other functions are not allowed to know the names of a function's *local variables*
- other functions are not allowed to know the *implementation details* of any other function

The general format of a function

```
return-value-type function-name (parameter-list)
{
    definitions
    statements
}
```

- return-value-type
 - the type of the value returned to the calling function
 - when no return value, use void
- parameter list
 - a comma separated list containing the definitions of the variables that will be passed to the function
 - when no argument, use void

Function Prototype

- declares the function's return type
- declares the number, types, and order of the parameters that the function expects to receive
- the function prototypes enable the compiler to verify that functions are called correctly
- the variable names in a function prototype are ignored
- implicit type conversion