Day07 A

Young W. Lim

2017-09-21 Thr

Outline

Based on

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

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
    - proivides 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`

Information Hiding

- 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