Modular Programming and Functions

Instructor: Chien-Ho Ko

Outlines

- Introduction
- Elements of module programs
- Program structure
- Return values under their names
- Functions with parameters
- Variable class/scope/visibility/duration
- Test drivers
- Style considerations

Introduction

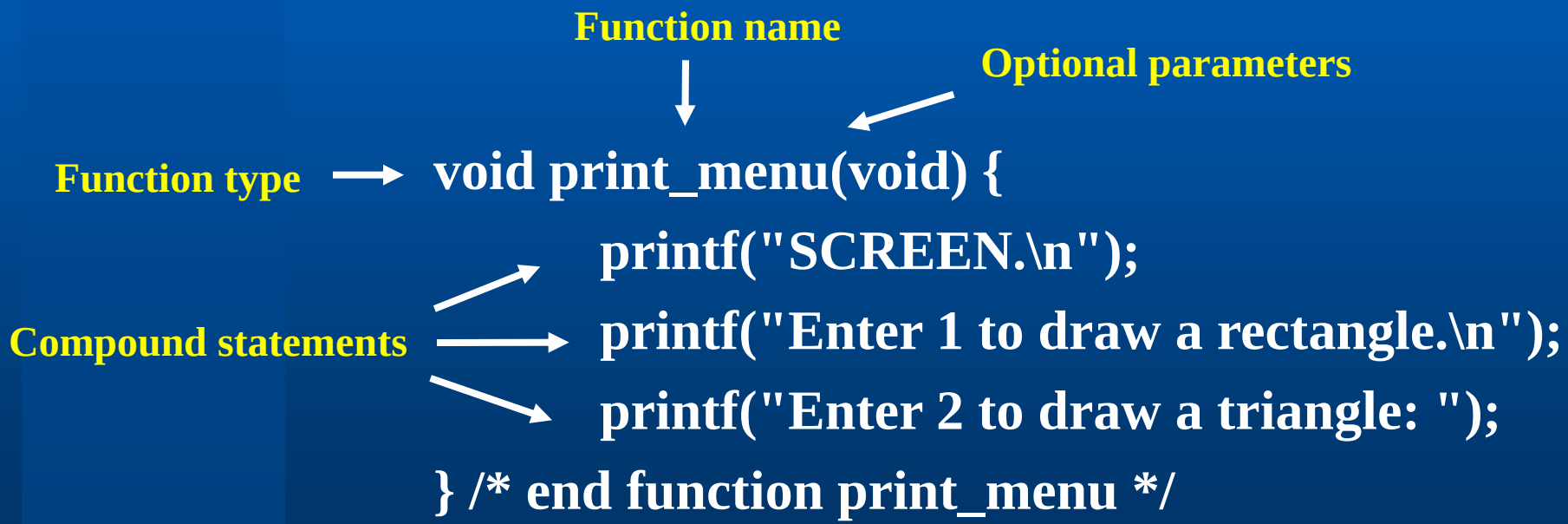
- **What is a modular?**
  - **Separately named and individually invokeable program elements**
- **What is modular programming?**
  - **Organize a program into small independent modules**
- **Its necessity**
  - **Development, debug, test, maintenance**
  - **Divide and conquer**

Elements of Module Programs (1/4)

- **Basics**
  - **Function name must be unique.**
  - **One of functions is the main function.**
    - **main**
  - **Three elements related to functions**
    - **Function Definition**
      - **void print_menu(void) { }**
    - **Function Calls**
      - **print_menu();**
      - **printf();**
    - **Function Declaration**
      - **void print_menu(void);**

Elements of Module Programs (2/4)

- **Function definition**



Elements of Module Programs (3/4)

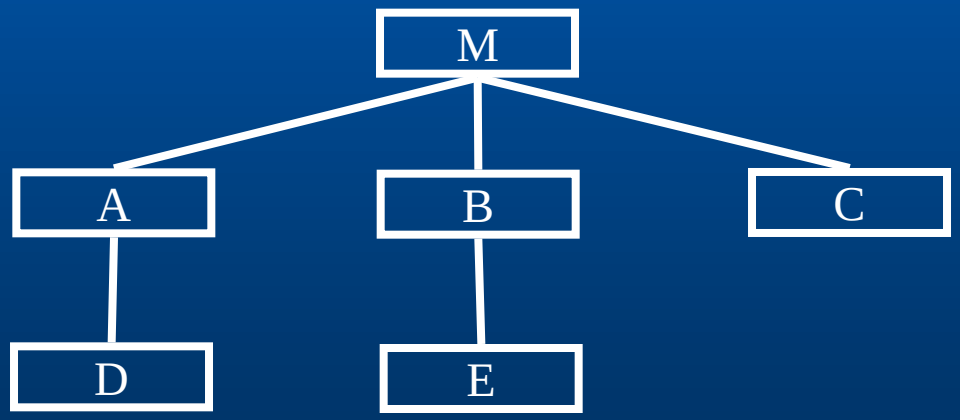
- **Function calls**
  - **Functions can be called by any functions.**
  - **Programmers defined functions cannot call main function.**
  - **print_menu();**

Elements of Module Programs (4/4)

- **Function declaration**
  - **Function prototype**
  - **void print_menu(void);**
  - **Global prototype**
    - **Outside function definitions**
  - **Local prototype**
    - **In a function definition**

Program Structure (1/2)

- **A collection of logical program modules describing invoked-by relationships.**

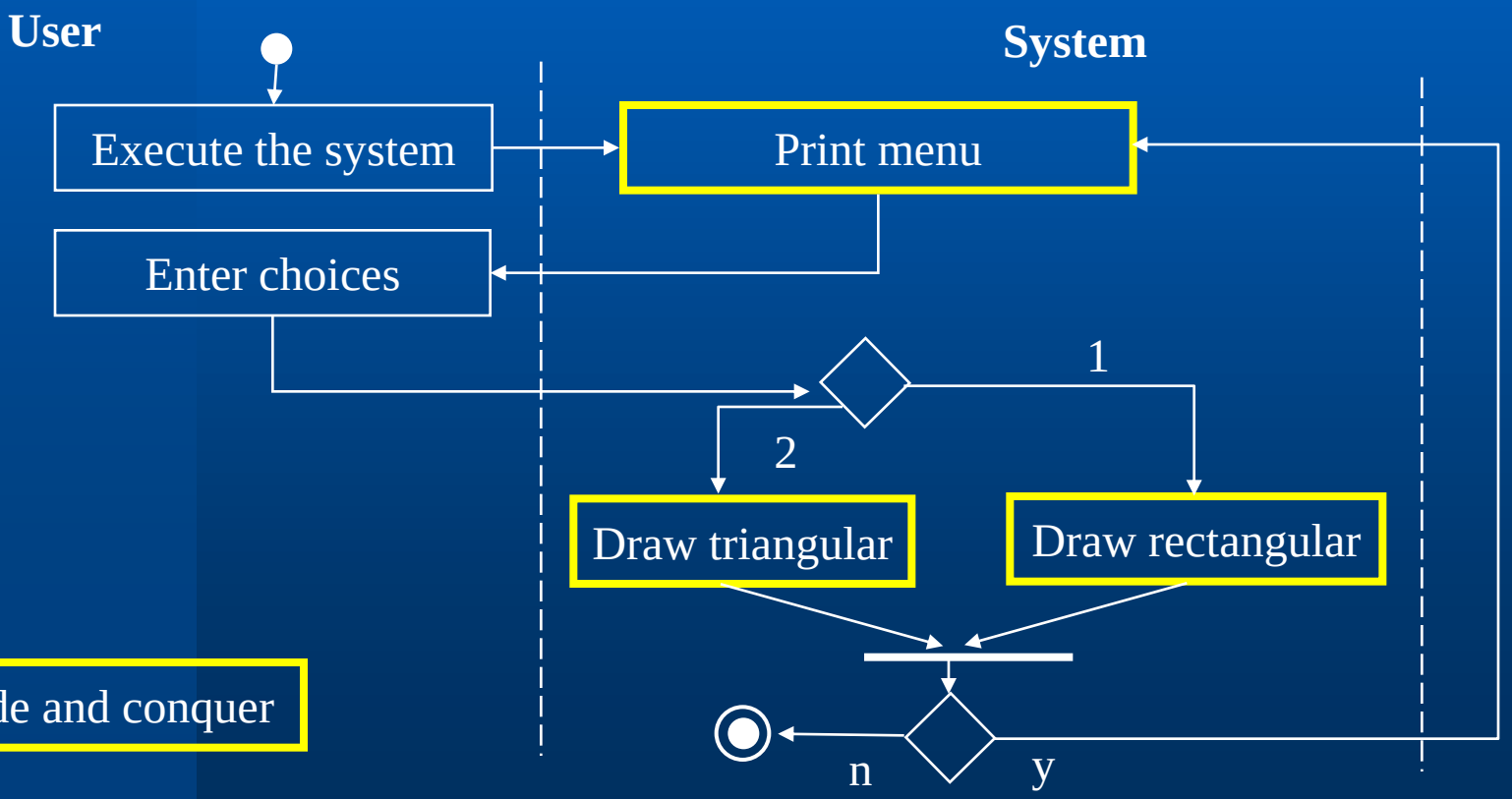


Program Structure (2/2)

- **Simple module invocation**
- **Conditional invocation**
- **Repetitive invocation**

Example 1 (1/2)

- **Swim lane diagram**



Divide and conquer

Example 1 (2/2)

- General C
- Function not received & return data
- Requirement specifications
  - Display a menu of choices
  - Input a request code
  - Print either a rectangular or a triangular
- 12_c01.c
- Exercise: add a module allowing users to enter their ID

Example 2

- **Robotic C**
- **Robot Go!!**
  - **If table is white, go forward; if table is black, rotate clockwise**
    - **Continue above selections.**
- **12_RobotC01.c**

Return Values Under Their Names

- **Value function**
  - **Produce and return one value**
  - **request_code=user_request();**
- **Return statement**
  - **Must include at least one *return* statement**
  - **The *return* terminates function execution**
  - **void function return;**

Example 3

- **Function return a single data**
- **Problem**
  - **Draw Geometrical Figures**
- **Requirement specifications**
  - **Display a menu of choices**
  - **Input a request code**
  - **Verify the request code**
  - **Print either a rectangular or a triangular**
- **12_c02.c**
- **Exercise: add a module allowing users to enter their ID. Then print it on the screen.**

Functions with Parameters (1/2)

- **Assign parameters through**
  - **Global variables (avoid)**
  - **Returned by a function**
  - **Using parameters**

Functions with Parameters (2/2)

- **Define function with parameters**
  - **void process_request(int request_value)**
- **Declare function with parameters**
  - **void process_request(int);**
- **Call function with parameters**
  - **process_request(process_code);**
  - **request_value=process_code**

Example 4

- **Functions with parameters**
- **General C**
- **Problem**
  - **Draw Geometrical Figures**
- **Requirement specifications**
  - **Display a menu of choices**
  - **Input a request code**
  - **Verify the request code**
  - **Print either a rectangular or a triangular**
- **12_c03.c**

Example 5

- **Robotic C**
- **Go Robot!**
- **12_RobotC02.c**

Variable Classes/Scope/Visibility/Duration

- **Visibility**
  - **Ability to access variable's memory location**
- **Duration/ life time**
  - **A time a memory location exist for that variable at run time**
  - **Static for global variable**
  - **Local for local variable**
  - **Dynamic by using special functions**

Test Drivers

- **Definition**
  - **A C program that calls on modules to verify whether modules work properly or not.**
- **Test correctness**
- **Examples**
  - **12_c04.c**
  - **12_RobotC03.c**

Style Considerations

- **2 to 50 Lines**
- **Function calls < 7**
- **Assign a descriptive name**
- **Use r*eturn* for a single value**
  - **Pointer for values**
- **Use available functions**
  - ***sqrt*()**