SUMMER SESSION II 2013
EECS 10 WEEK4 DISCUSSION1

Che-Wei Chang

OUTLINE

- Reminder: 2nd midterm is coming.
- Assignment 4 Part1
  - Menu Driven Calculator for floating point number [25pts]
- Concept review: Function

Assignment Discussion

- Assignment 4, Part 1
  - Before you implement your work, review lecture slides about function declaration, function definition, and function call.
  - Read the assignment handout carefully
- Menu driven calculator for floating point number
  - Good exercise for function call
  - What is the input? What is the output?
  - What algorithm to solve this problem?
  - What is the control flow for this program?
  - How to implement this program?

Menu Driven Calculator

- Prompt a menu and user can choose the operation
- Operation List
  - 1. Add
  - 2. Subtract
  - 3. Multiply
  - 4. Divide
  - 5. Absolute
  - 6. Square Root
  - 7. sine
  - 8. cosine
  - 9. tangent
  - 10. N-th root (Bonus)
  - 11. Quit

Part 1

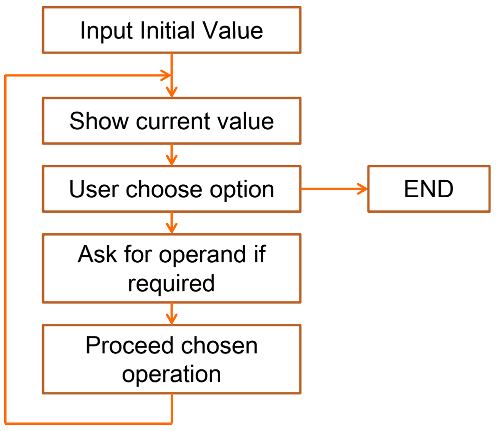
Part 2

Menu Driven Calculator

- Should be aware of the following points:
  - The execution only finishes when the user want to quit
    - Infinite loop, with breaking condition
  - Asking for additional operand if required
    - Ex: user chooses addition → asking for the 2nd operand
    - Ex: user chooses absolute → 2nd operand is not needed.
  - A function has to be created for the corresponding operation
    - Ex:
```
/*function Add for add operation*/
double Add(double op1, double op2)
```
  - Misuse of the calculator must be handled
    - Notify the user with error message when the input is invalid
    - Ex: report `"ERROR: Division by zero"` when user input 0 as the divisor.

Menu Driven Calculator

- Flow:



CONCEPT REVIEW: FUNCTION

- Important programming concepts
- C programming language distinguishes 3 constructs around functions:
  - Function declaration
    - Declaration of function name, parameters, and return type.
  - Function definition
    - An implicit function declaration
    - Extension of a function declaration with a function body
    - Function declaration + function behavior
  - Function call
    - Invocation of a function
    - Supply argument for formal parameters

C PROGRAM RULES

- Rules for function
  - A function must be declared before it can be called
  - Multiple function declarations are allowed (if they match)
  - A function declaration is an implicit function declaration
  - A function must be defined exactly once in a program
  - A function can be called for any number of times
- Scope of an identifier
  - Portion of the program where the identifier can be referred
  - Global variables *file scope*
  - Function parameter *function scope*
  - Local parameter *block scope*

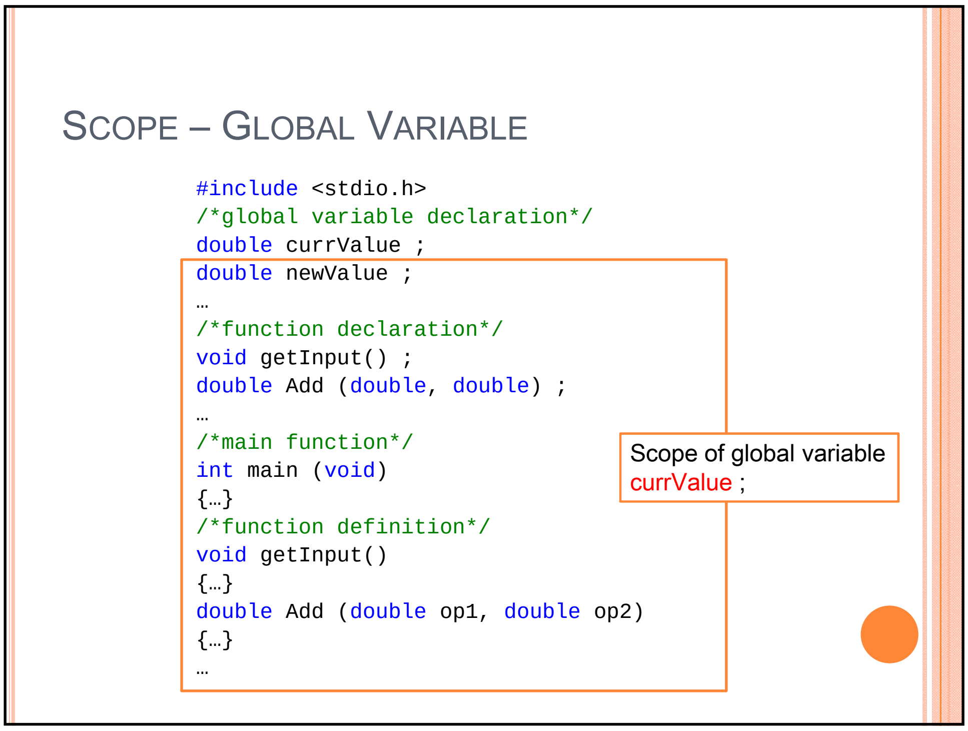
Function – Use calculator as the example

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function declaration*/
void getInput() ;
double Add (double, double) ;
…
/*main function*/
int main (void)
{…}
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
```

Scope – Global Function

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function declaration*/
void getInput() ;
double Add (double, double) ;
…
/*main function*/
int main (void)
{…}
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
```

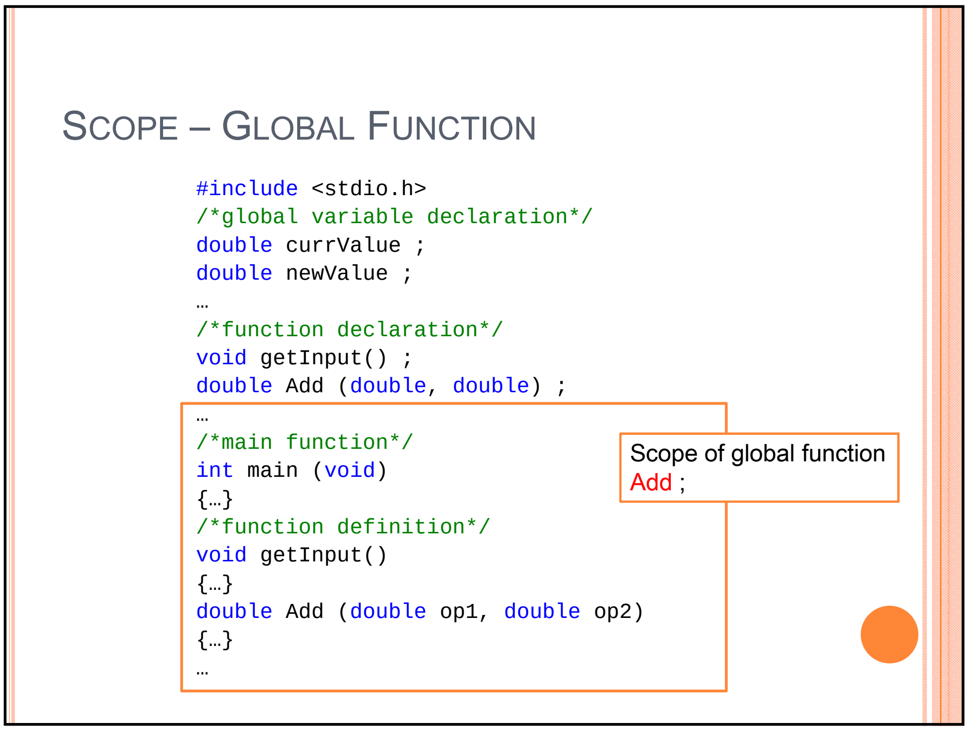
Scope of global function
printf(), scanf(), etc.



Scope – Global Variable

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function declaration*/
void getInput() ;
double Add (double, double) ;
…
/*main function*/
int main (void)
{…}
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
```

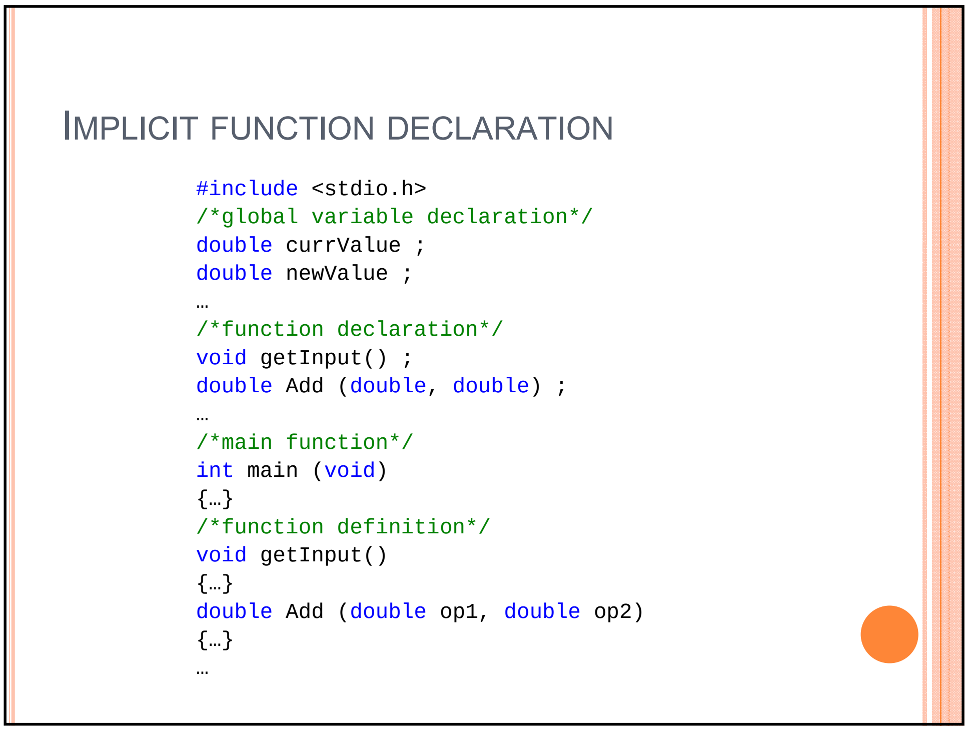
Scope of global variable currValue ;



Scope – Global Function

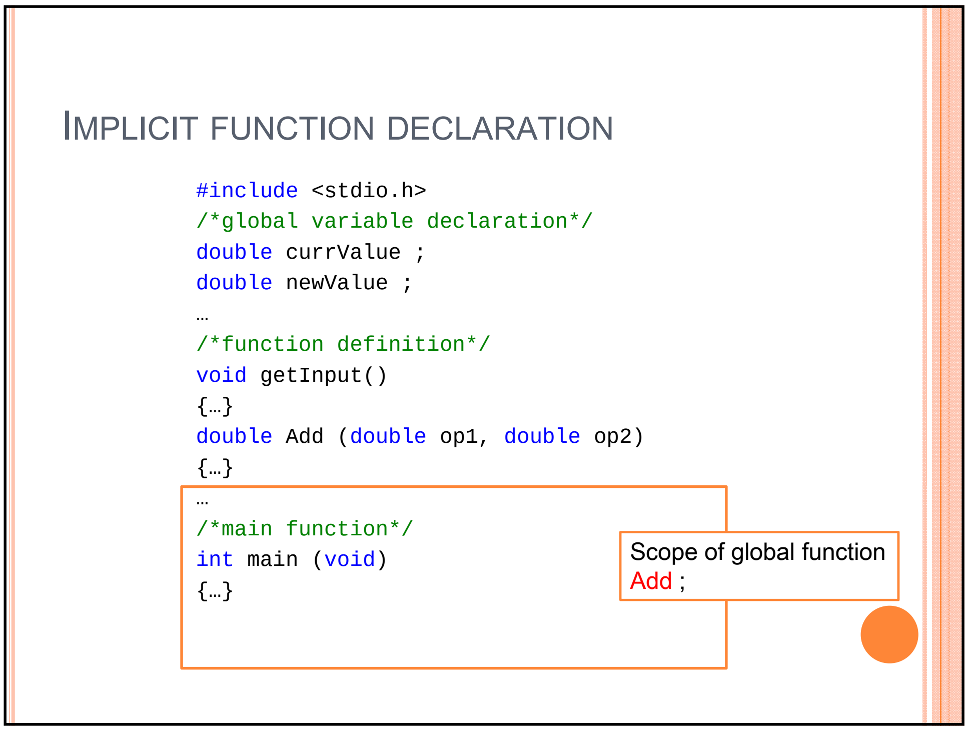
```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function declaration*/
void getInput() ;
double Add (double, double) ;
…
/*main function*/
int main (void)
{…}
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
```

Scope of global function Add ;



Implicit function declaration

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function declaration*/
void getInput() ;
double Add (double, double) ;
…
/*main function*/
int main (void)
{…}
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
```



Implicit function declaration

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
/*main function*/
int main (void)
{…}
```

Scope of global function Add ;

Implicit function declaration

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*main function*/
int main (void)
{…}
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{…}
…
```

Compiler will complain

Scope of global function Add ;

Scope – Local parameter

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{
  double sum ;
  sum = op1 + op2 ;
  return sum ;
}
/*main function*/
int main (void)
{…}
```

Scope of parameter op1 ;

Scope – Local variable

```
#include <stdio.h>
/*global variable declaration*/
double currValue ;
double newValue ;
…
/*function definition*/
void getInput()
{…}
double Add (double op1, double op2)
{
  double sum ;
  sum = op1 + op2 ;
  return sum ;
}
/*main function*/
int main (void)
{…}
```

Scope of parameter sum ;

Goal Today

- Finish the implementation of menu-driven control flow
- Finish the implementation of the first five options.
- Part 2 assignment will be built on the part 1.