Jiangxi University of Science and Technology

Ch06 Modularity Using Functions: Part I

Lecture0601 Function and Parameter Declarations

Objectives

- 6.1 Function and Parameter Declarations
- 6.2 Returning a Value
- 6.3 Case Study: Calculating Age Norms
- 6.4 Standard Library Functions
- 6.5 Common Programming and Compiler Errors

6.1 Function and Parameter Declarations

- A function that is called into action by its reference in another function is *a* ***called function*** **被调函数**
- A function that calls another function is referred to as the ***calling function*** **主调函数**

$$\underbrace{functionName}_{\text{This identifies the called function}}\ \underbrace{(data\ passed\ to\ function)}_{\text{This passes data to the function}}\ ;$$

Figure 6.1 Calling and passing data to a function

6.1 Function and Parameter Declarations

➢ Program 6.1

```
#include <stdio.h>
int main(){
    void findMax(float, float); //the function prototype
    float firstnum, secnum;
    printf("Enter a number: ");
    scanf("%f", &firstnum);
    printf("Great! Please enter a second number: ");
    scanf("%f", &secnum);
    findMax(firstnum, secnum); //the function is called here
    return 0;
}
```

```
//the following is the function findMax
void findMax(float x, float y)//this is the function's header line
{
    float maxnum;
    if (x >= y) //find the maximum number
        maxnum = x;
    else
        maxnum = y;
printf("\nThe maximum of the two numbers entered is %f\n", maxnum);
}
```

TEST Me!

6.1 Function and Parameter Declarations

➢ 1. Function Prototypes 函数原型

- The ***declaration statement*** for a function is referred to as a function prototype
- ***returnDataType functionName(argument data types);***

① **returnDataType** Declares the data type of the value that will be directly returned by the function

② **argument data types** Declares the data type of the values that need to be transmitted to the called function when it is invoked

- e.g. **void findMax(float, float);**

6.1 Function and Parameter Declarations

➢ 1. Function Prototypes (函数原型)

- Function prototypes allow the compiler to check for data type errors
- If the function prototype does not agree with data types specified when the function is written, an error message (typically **TYPE MISMATCH 类型不匹配**) will occur

6.1 Function and Parameter Declarations

➢ Calling a Function 调用函数

- items enclosed in parentheses in a function **call statement 调用语句** are ***actual arguments*** **(实参)**and ***actual parameters***
- ***Pass by value*****(传值)**: when a function ***receives copies of the values*** in each argument and must determine where to store them before it does anything else
- Also referred to as call by value

6.1 Function and Parameter Declarations

- Calling and passing two values to Findmax()

```
#include <stdio.h>
int main(){
    void findMax(float , float ); //the function prototype
    float firstnum, secnum;
    printf("Enter a number: ");
    scanf("%f", &firstnum);
    printf("Great! Please enter a second number: ");
```

C:\Users\Chengtian Ouyang\Documents\Visual Studio 2010\Projects\qq\Debug\qq.exe

```
Enter a number:
```

6.1 Function and Parameter Declarations

➢ Function Header Line

- ***Function header 函数首部***: identifies the data type of the return value, provides the function with a name, and specifies the number, order, and type of values expected by the function
- ***Function body 函数体***: operates on the passed data and returns, at most, one value
- The argument names in the header line are known as parameters or ***formal parameters*** and ***formal arguments 形参***

6.1 Function and Parameter Declarations

➢Function Header Line

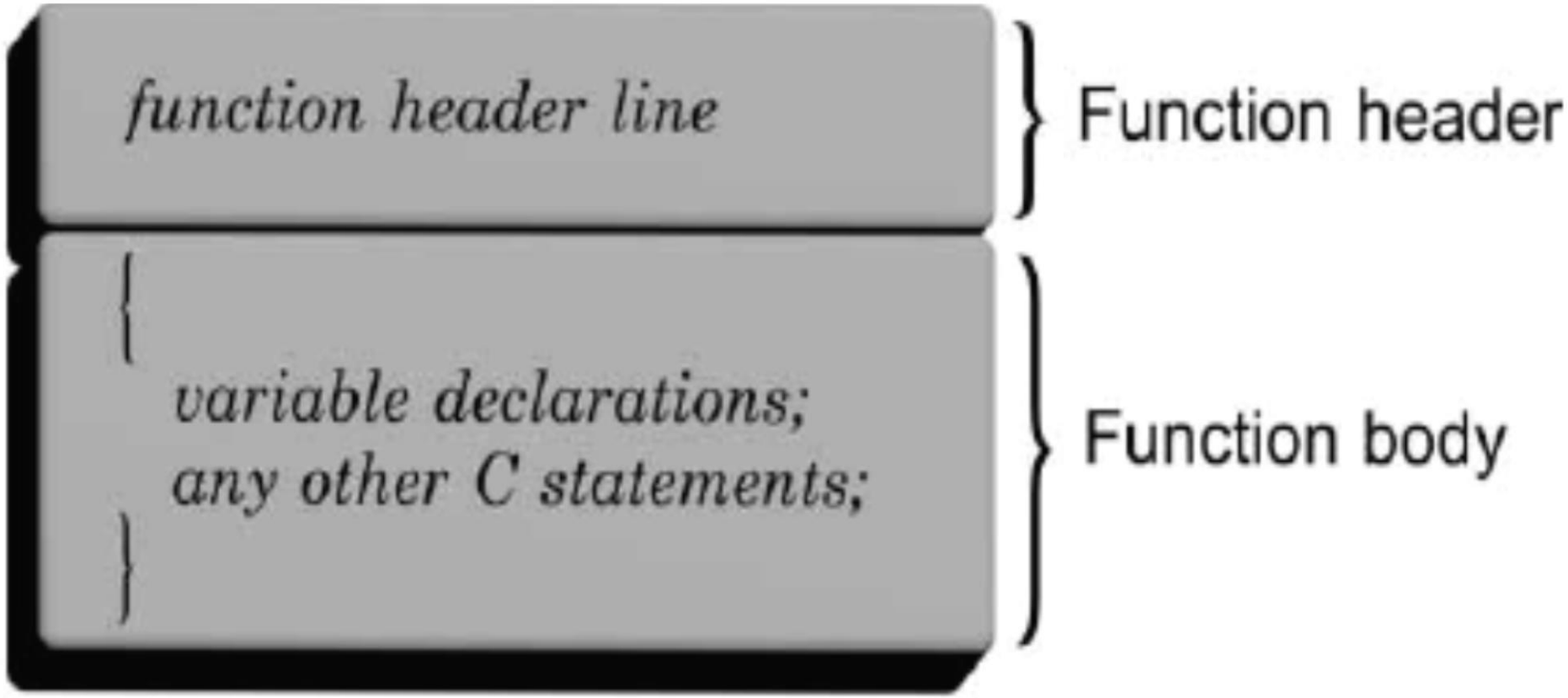

Figure 6.4 General format of a function

6.1 Function and Parameter Declarations

➢Function Header Line

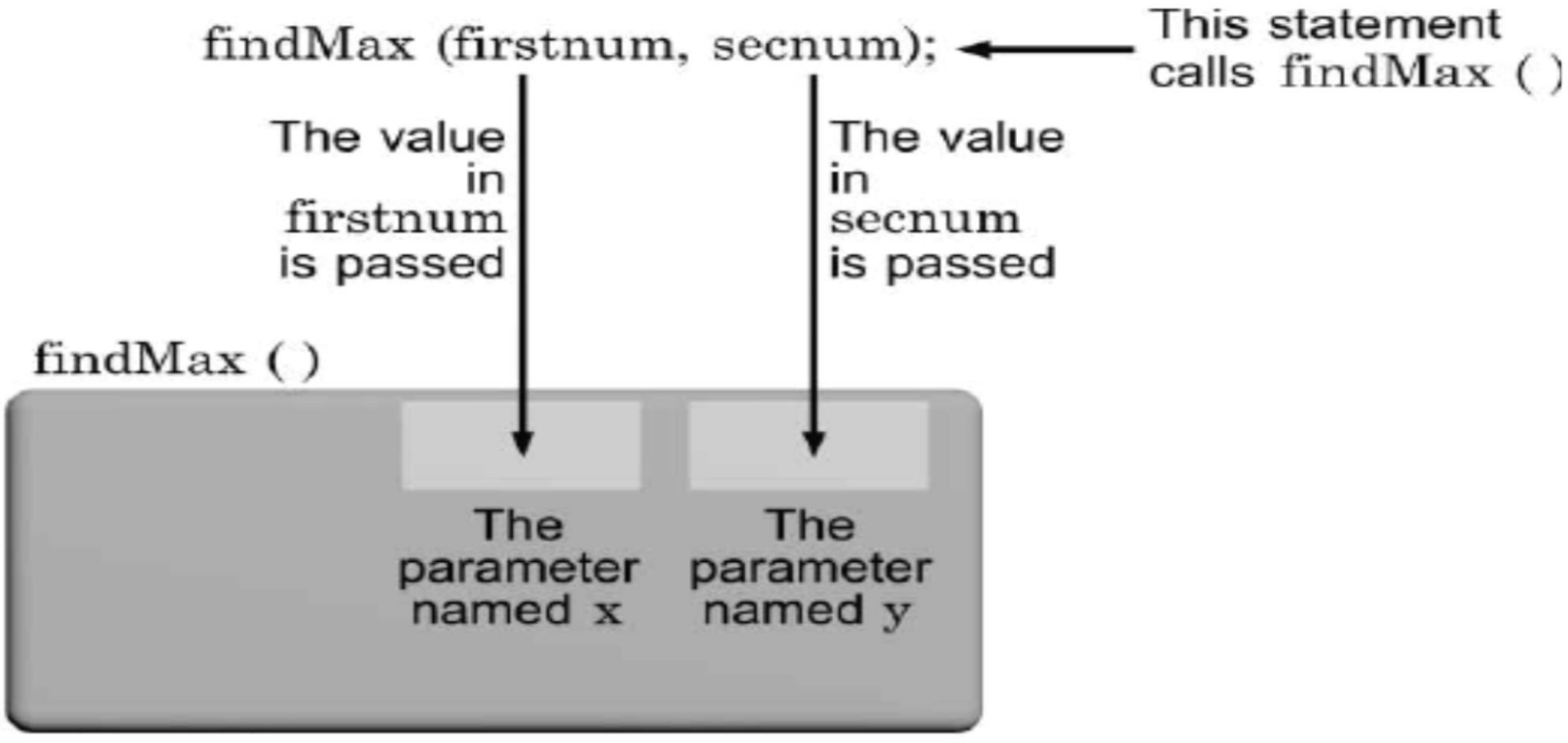

Figure 6.5 Storing values into parameters

6.1 Function and Parameter Declarations

➢ Function Header Line

- ***main()*** must adhere to the rules required for constructing all C functions
- Some programmers prefer to put all called functions at the top of a program and make ***main()*** the last function listed
- Each C function is a separate and independent entity with its own parameters and variables
- Nested functions are not permitted
- The function's prototype, along with pre- and postconditions should provide all the information necessary to call the function successfully

➢ Placement of Statements

- All preprocessor directives, variables, named constants, and functions, except main(), must be either declared or defined before they can be used

6.1 Function and Parameter Declarations

- Basic (good) programming structure:
  - **preprocessor directives**
  - **symbolic constants**
  - **function prototypes can be placed here**
  - **int main()**
  - **{**
  - **function prototypes can be placed here**
  - **variable declarations;**
  - **other executable statements;**
  - **return value;**
  - **}**

6.1 Function and Parameter Declarations

Program 6.3

```c
#include <stdio.h>//preprocessor directives
int main(){
#define MAXCOUNT 4//symbolic constants
    void tempConvert(float ); //function prototype
    int count;  //start of variable declarations
    float fahren;
    for(count = 1; count <= MAXCOUNT; count++){
        printf("Enter a Fahrenheit temperature: ");
        scanf("%f", &fahren);
        tempConvert(fahren);
    }
    return 0;
}
void tempConvert(float inTemp) /* function header */
{
printf("The Celsius equivalent is %6.2f\n",(5.0/9.0)*(inTemp-32.0) );
}
```

Reference

- BOOK
- Some part of this PPT given by Prof 欧阳城添

(Prof: Chengtian Ouyang)

- with special thank
- https://www.codingunit.com/c-tutorial-first-c-program-hello-world