COMSATS University
Islamabad
(Lahore Campus)

CSC103- Programming Fundamentals

MS. MAHWISH WAQAS

MAWISH.WAQAS@CUILAHORE.EDU.PK

Chapter 6:
User-Defined Functions

Objectives (cont'd.)

- In this chapter, you will:
  - Learn how to construct and use `void` functions
  - Discover the difference between value and reference parameters
  - Explore reference parameters and value-returning functions
  - Learn about the scope of an identifier
  - Examine the difference between local and global identifiers
  - Discover static variables

Objectives (cont'd.)

- Learn how to debug programs using drivers and stubs
- Learn function overloading
- Explore functions with default parameters

Void Functions

- User-defined void functions can be placed either before or after the function `main`
- If user-defined void functions are placed after the function `main`
  - The function prototype must be placed before the function `main`
- Void function does not have a return type
  - `return` statement without any value is typically used to exit the function early *prematurely*

Void Functions (cont'd.)

- Formal parameters are optional
- A call to a void function is a *stand-alone* statement
  - Cannot use function call with any other expression or in a cout statement
- Void function definition syntax:

```
void functionName(formal parameter list)
{
     statements
}
```

Void Functions (cont'd.)

- Formal parameter list syntax:

```
dataType& variable, dataType& variable, ...
```

- Function call syntax:

```
functionName(actual parameter list);
```

- Actual parameter list syntax:

```
expression or variable, expression or variable, ...
```

Void Functions (cont'd.)

```
void funexp(int a, double b, char c, int x)
{
    .
    .
    .
}
```

The function `funexp` has four parameters.

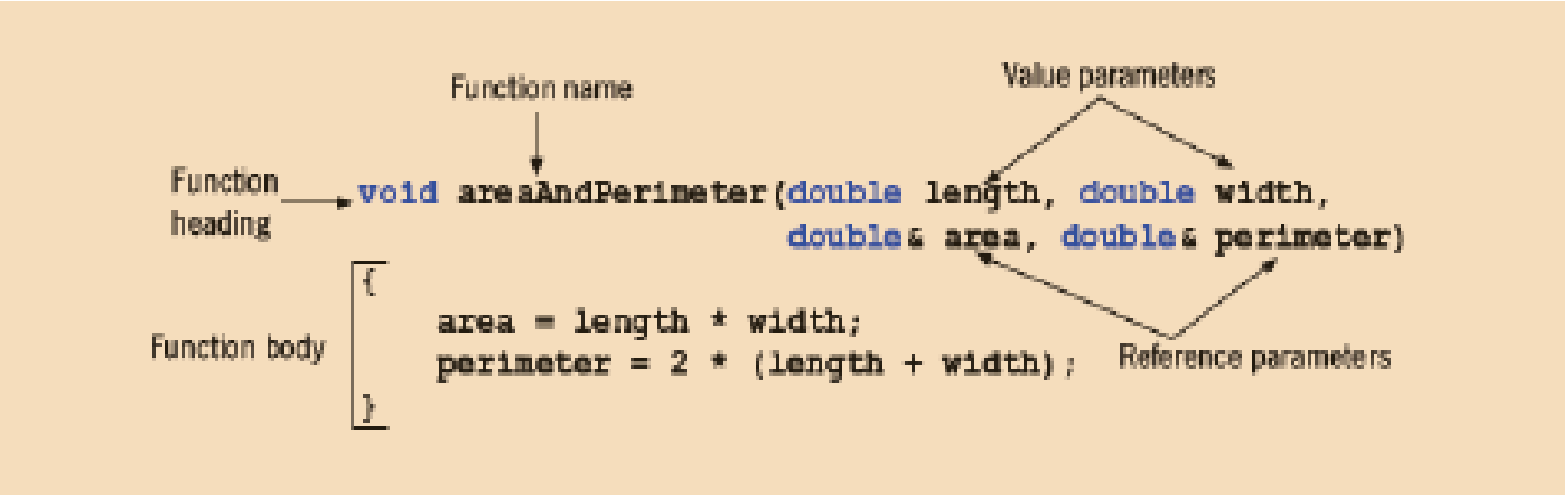
Void Functions (cont'd.)

- Value parameter: a formal parameter that receives a copy of the content of corresponding actual parameter
- Reference parameter: a formal parameter that receives the location (memory address) of the corresponding actual parameter

```
void areaAndPerimeter(double length, double width,
                      double& area, double& perimeter)
{
    area = length * width;
    perimeter = 2 * (length + width);
}
```

The function **areaAndPerimeter** has four parameters: **length** and **width** are value parameters of type double; and **area** and **perimeter** are reference parameters of type double.

Figure 6-4 describes various parts of the function **areaAndPerimeter**.



Value Parameters

- If a formal parameter is a value parameter:
  - The value of the corresponding actual parameter is copied into it
  - Formal parameter has its own copy of the data
- During program execution
  - Formal parameter manipulates the data stored in its own memory space

```
void funcValueParam(int num);                                   //Line 3

int main()                                                      //Line 4
{                                                               //Line 5
    int number = 6;                                             //Line 6

    cout << "Line 7: Before calling the function "
         << "funcValueParam, number = " << number
         << endl;                                               //Line 7

    funcValueParam(number);                                     //Line 8

    cout << "Line 9: After calling the function "
         << "funcValueParam, number = " << number
         << endl;                                               //Line 9

    return 0;                                                   //Line 10
}                                                               //Line 11

void funcValueParam(int num)                                    //Line 12
{                                                               //Line 13
    cout << "Line 14: In the function funcValueParam, "
         << "before changing, num = " << num
         << endl;                                               //Line 14

    num = 15;                                                   //Line 15

    cout << "Line 16: In the function funcValueParam, "
         << "after changing, num = " << num
         << endl;                                               //Line 16
}                                                               //Line 17
```

Sample Run:

```
Line 7: Before calling the function funcValueParam, number = 6
Line 14: In the function funcValueParam, before changing, num = 6
Line 16: In the function funcValueParam, after changing, num = 15
Line 9: After calling the function funcValueParam, number = 6
```

Reference Variables as Parameters

- If a formal parameter is a reference parameter
  - It receives the memory address of the corresponding actual parameter
- During program execution to manipulate data
  - Changes to formal parameter will change the corresponding actual parameter

Reference Variables as Parameters (cont'd.)

- Reference parameters are useful in three situations:
  - Returning more than one value
  - Changing the actual parameter
  - When passing the address would save memory space and time

Value and Reference Parameters and Memory Allocation

- When a function is called
  - Memory for its formal parameters and its local variables is allocated in the function data area
- For a value parameter, the actual parameter's value is copied into the formal parameter's memory cell
  - Changes to the formal parameter do not affect the actual parameter's value

Value and Reference Parameters and Memory Allocation (cont'd.)

- For a reference parameter, the actual parameter's address passes to the formal parameter
  - Both formal and actual parameters refer to the same memory location
  - During execution, changes made to the formal parameter's value permanently change the actual parameter's value

Reference Parameters and Value-Returning Functions

- Can also use reference parameters in a value-returning function
  - Not recommended
- By definition, a value-returning function returns a single value via `return` statement
- If a function needs to return more than one value, change it to a void function and use reference parameters to return the values

Reference Parameters Example

Example 6-14, page # 391.