

# CHAPTER 4

# FUNCTIONS

Dr. Shady Yehia Elmashad



# Outline

- 1. Introduction**
- 2. Program Components in C++**
- 3. Math Library Functions**
- 4. Functions**
- 5. Function Definitions**
- 6. Function Prototypes**
- 7. Header Files**
- 8. Default Arguments**

# 1. Introduction

- Divide and conquer
  - Construct a program from smaller pieces or components.
  - Each piece more manageable than the original program.

## 2. Program Components in C++

- Programs written by
  - combining new functions with “prepackaged” functions in the C++ standard library.
  - The standard library provides a rich collection of functions.
- Functions are invoked by a function call
  - A function call specifies the function name and provides information (as arguments) that the called function needs
  - Boss to worker analogy:
    - A boss (the calling function or caller) asks a worker (the called function) to perform a task and return (i.e., report back) the results when the task is done.*

## 2. Program Components in C++

- **Function definitions**

- Only written once
- These statements are hidden from other functions.
- Boss to worker analogy:

*The boss does not know how the worker gets the job done; he just wants it done*

# 3. Math Library Functions

- Math library functions

- Allow the programmer to perform common mathematical calculations
- Are used by including the header file `<cmath>`

- Functions called by writing

*functionName (argument)*

- Example

```
cout << sqrt( 900.0 );
```

- Calls the **sqrt** (square root) function. The preceding statement would print **30**
- The **sqrt** function takes an argument of type **double** and returns a result of type **double**, as do all functions in the math library

# 3. Math Library Functions

- Function arguments can be

- Constants

```
sqrt( 4 );
```

- Variables

```
sqrt( x );
```

- Expressions

```
sqrt( sqrt( x ) );
```

```
sqrt( 3 - 6x );
```

# 4. Functions

- **Functions**
  - Allow the programmer to modularize a program
- **Local variables**
  - Known only in the function in which they are defined
  - All variables declared in function definitions are local variables
- **Parameters**
  - Local variables passed when the function is called that provide the function with outside information



# 5. Function Definitions

- Create customized functions to
  - Take in data
  - Perform operations
  - Return the result
- Format for function definition:

```
return-value-type function-name( parameter-list )  
  {  
    declarations and statements  
  }
```

- Example:

```
int square( int y)  
{  
    return y * y;  
}
```

**1. Function prototype****2. Loop****3. Function definition**

```
1 // Fig. 3.3: fig03_03.cpp
2 // Creating and using a programmer-defined function
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 int square( int ); // function prototype
9
10 int main()
11 {
12     for ( int x = 1; x <= 10; x++ )
13         cout << square( x ) << " ";
14
15     cout << endl;
16     return 0;
17 }
18
19 // Function definition
20 int square( int y )
21 {
22     return y * y;
23 }
```

Notice how parameters and return value are declared.

```
1  4  9 16 25 36 49 64 81 100
```

**Program Output**

## Outline



### 1. Function prototype (3 parameters)

### 2. Input values

### 2.1 Call function

```
1 // Fig. 3.4: fig03_04.cpp
2 // Finding the maximum of three integers
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 int maximum( int, int, int ); // function prototype
10
11 int main()
12 {
13     int a, b, c;
14
15     cout << "Enter three integers: ";
16     cin >> a >> b >> c;
17
18     // a, b and c below are arguments to
19     // the maximum function call
20     cout << "Maximum is: " << maximum( a, b, c ) << endl;
```

### 3. Function definition

```
21
22     return 0;
23 }
24
25 // Function maximum definition
26 // x, y and z below are parameters to
27 // the maximum function definition
28 int maximum( int x, int y, int z )
29 {
30     int max = x;
31
32     if ( y > max )
33         max = y;
34
35     if ( z > max )
36         max = z;
37
38     return max;
39 }
```

```
Enter three integers: 22 85 17
Maximum is: 85
```

```
Enter three integers: 92 35 14
Maximum is: 92
```

```
Enter three integers: 45 19 98
Maximum is: 98
```

### Program Output

# 6. Function Prototypes (declaration)

- Function prototype

- Function name

- Parameters

  - Information the function takes in

- Return type

  - Type of information the function passes back to caller (default **int**)

  - void** signifies the function returns nothing

- Only needed if function definition comes after the function call in the program

- Example:

```
int maximum( int, int, int );
```

- Takes in 3 **ints**

- Returns an **int**

# 7. Header Files

- Header files

- Contain function prototypes for library functions
- `<cstdlib>`, `<cmath>`, etc.
- Load with `#include <filename>`

- Example:

```
#include <cmath>
```

- Custom header files

- Defined by the programmer
- Save as `filename.h`
- Loaded into program using

```
#include "filename.h"
```

# Room Area (Rectangle)

```
# include < iostream.h >
float findArea ( float ,float ) ;           // Function declaration (prototype)
void main ( ) {
float room_L, room_W, room_Area;
cout << " Please enter the room width " ;
cin >> room_W ;
cout << " Please enter the room length " ;
cin >> room_L ;

room_Area = findArea (room_W, room_L );    // Function call

cout << " The area of your room is : " << room_Area << " square unit " ;
}

float findArea ( L , W ) {                // Function definition
float area;
Area = L * W ;
return Area ;
}
```

# Celsius to Fahrenheit Temperature Converter

```
# include < iostream.h >
float convert ( float ) ;           // Function declaration (prototype)
void main ( ) {
float Temp_Fah, Temp_Cen;
cout << " Please enter the temperature in fahrenheit " ;
cin >> Temp_Fah ;

Temp_Cen = convert ( Temp_Fah ) ;  // Function call

cout << " The temperature in centigrade is: :" << Temp_Cen ;
}

float covert ( float Fah ) {       // Function definition
float Cen;
Cen = (Fah - 32 ) * (5 / 9 );
return Cen ;
}
```



# Odd or Even

```
# include < iostream.h >
```

```
void odd_even ( int ) ;           // Function declaration (prototype)
```

```
void main ( ) {
```

```
int number ;
```

```
cout << " Please enter a number: " ;
```

```
cin >> number ;
```

```
odd_even ( number ) ;           // Function call
```

```
}
```

```
void odd_even ( int number ) {   // Function definition
```

```
if ( number % 2 == 0 )
```

```
cout << " you number is even " ;
```

```
else
```

```
cout << " you number is odd " ;
```

```
}
```

# Positive or Negative

```
# include < iostream.h >
```

```
void poitive_negative ( int );           // Function declaration (prototype)
```

```
void main ( ) {
```

```
int number ;
```

```
cout << " Please enter a number: " ;
```

```
cin >> number ;
```

```
poitive_negative ( number );           // Function call
```

```
}
```

```
void poitive_negative ( int number ) {           // Function definition
```

```
if ( number > 0 )
```

```
cout << " you number is positive " ;
```

```
else
```

```
cout << " you number is negative " ;
```

```
}
```

# Swap

```
# include < iostream.h >
void Swap( int , int ) ;           // Function declaration (prototype)
void main ( ) {
int n1, n2 ;
cout << " Please enter the value of number 1" ;
cin >> n1 ;
cout << " Please enter the value of number 2" ;
cin >> n2 ;

Swap ( n1, n2 ) ;   // Function call
}

void Swap ( int n1 , int n2 ) {    // Function definition
int temp ;
temp = n1;
n1 = n2;
n2 = temp;
cout << " The value stored in number 1 is now: " << n1 << endl ;
cout << " The value stored in number 2 is now: " << n2 << endl ;
}
```

# Product and Quotient of two numbers

```
# include < iostream.h >
float Product ( float , float ) ;           // Function declaration (prototype)
float Quotient( float , float ) ;         // Function declaration (prototype)
void main ( ) {
int n1, n2 ;
cout << " Please enter two numbers " ;
cin >> a >> b ;

R1 = Product (a, b);                       // Function call
R2 = Quotient (a, b);                      // Function call
cout << "the product of them is" << R1 << " and the division is" << R2;
}

float Product ( float a , float b ) {     // Function definition
return a*b;
}

float Quotient ( float a , float b ) {    // Function definition
If (b != 0) {
float Q = a / b;
return Q; }
else cout << " You couldn't divide by zero. " ;
}
```

# Draw the following Pattern

```
#include <iostream.h >
void draw_line ( void );
void main ( ) {
```

```
draw_line ( );
draw_line ( );
cout << " Welcome " << endl ;
draw_line ( );
cout << " First Year " ;
draw_line ( );
draw_line ( );
}
```

```
void draw_line ( void ) {
for ( int i = 0 ; i < 5 ; i ++ ) {
cout << " * " ; }
cout << endl ;
}
```

```
// Function declaration (prototype)
```

```
// Function call
```

```
// Function call
```

```
// Function call
```

```
// Function call
```

```
// Function call
```

```
// Function definition
```

```
* * * * *
```

```
* * * * *
```

```
Welcome
```

```
* * * * *
```

```
First Year
```

```
* * * * *
```

```
* * * * *
```

# 8. Default Arguments

- If function parameter omitted, gets default value
  - Can be constants, global variables, or function calls
  - If not enough parameters specified, rightmost go to their defaults
- Set defaults in function prototype

```
int defaultFunction( int x = 1,  
                    int y = 2, int z = 3 );
```

## Outline



### 1. Function prototype

### 2. Print default volume

#### 2.1 Print volume with one parameter

#### 2.2 Print with 2 parameters

#### 2.3 Print with all parameters.

### 3. Function definition

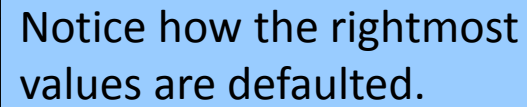
```
1 // Fig. 3.23: fig03_23.cpp
2 // Using default arguments
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 int boxVolume( int length = 1, int width = 1, int height = 1 );
9
10 int main()
11 {
12     cout << "The default box volume is: " << boxVolume()
13         << "\n\nThe volume of a box with length 10,\n"
14         << "width 1 and height 1 is: " << boxVolume( 10 )
15         << "\n\nThe volume of a box with length 10,\n"
16         << "width 5 and height 1 is: " << boxVolume( 10, 5 )
17         << "\n\nThe volume of a box with length 10,\n"
18         << "width 5 and height 2 is: " << boxVolume( 10, 5, 2 )
19         << endl;
20
21     return 0;
22 }
23
24 // Calculate the volume of a box
25 int boxVolume( int length, int width, int height )
26 {
27     return length * width * height;
28 }
```

```
The default box volume is: 1
```

```
The volume of a box with length 10,  
width 1 and height 1 is: 10
```

```
The volume of a box with length 10,  
width 5 and height 1 is: 50
```

```
The volume of a box with length 10,  
width 5 and height 2 is: 100
```



Notice how the rightmost values are defaulted.



# Example: What is the O/P?

```
# include < iostream.h >
int number = 10 ;
void display ( void ) ;
void main ( ) {
int number = 20 ;
cout << " The value of the number is " << number << endl ;
display ( ) ;
}
void display ( void ) {
cout << " The value of the number now is " << number ;
}
```

Output:           The value of the number is 20  
                  The value of the number now is 10