National University of computer and emerging sciences

Foundation for Advancement of Science and Technology

Programming Fundamentals

Function in C++

BS (SE) Fall-2025

Tasks

- Write a function isEven(int n) that returns true if n is even, otherwise false.
- Write a function power(int base, int exponent) that returns the result of base^exponent.
- Write a function reverseNumber(int n) that returns the reverse of a number.
- Create a function countVowels(string s) that counts the number of vowels in a string.

Agenda

- Function overview
- Modular programming
- Defining and calling functions
- Function prototype (declaration)

Function overview

- Function is a set of instruction that are designed to perform a specific task
  - Function is a complete and independent program
  - Divide a large program into smaller units
  - User defined and built-in functions
- Modular programming:
  - breaking a program up into smaller, manageable functions or modules
- Some advantages of functions
  - Functions makes a program clear and understandable
  - Finding errors will be easily (easy to debug)
  - Avoids code repetition and saves development time (code reusability)
  - Makes a program modification easy without changing the structure of a program (easy code maintenance)
  - Improves maintainability of programs
  - Simplifies the process of writing programs

Example

This program has one long, complex function containing all of the statements necessary to solve a problem.

```
int main()
{
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
}
```

In this program the problem has been divided into smaller problems, each of which is handled by a separate function.

```
int main()
{
   statement;
   statement;
   statement;
}
```

main function

```
void function2()
{
   statement;
   statement;
   statement;
}
```

function 2

```
void function3()
{
   statement;
   statement;
   statement;
}
```

function 3

```
void function4()
{
   statement;
   statement;
   statement;
}
```

function 4

User define functions

- Functions created by user as part of the program
- These functions are used for a specific use/purpose
- User defined function has three parts
  - Function declaration
  - Function definition
  - Function calling

Defining and Calling Functions

- Function call: statement causes a function to execute
- Function definition: statements that make up a function

Function Definition

- Definition includes: (Declarator and Body of Function)
  - return type: data type of the value that function returns to the part of the program that called it
  - name: name of the function. Function names follow same rules as variables
  - parameter list: variables containing values passed to the function
  - body: statements that perform the function's task, enclosed in { }

Example program

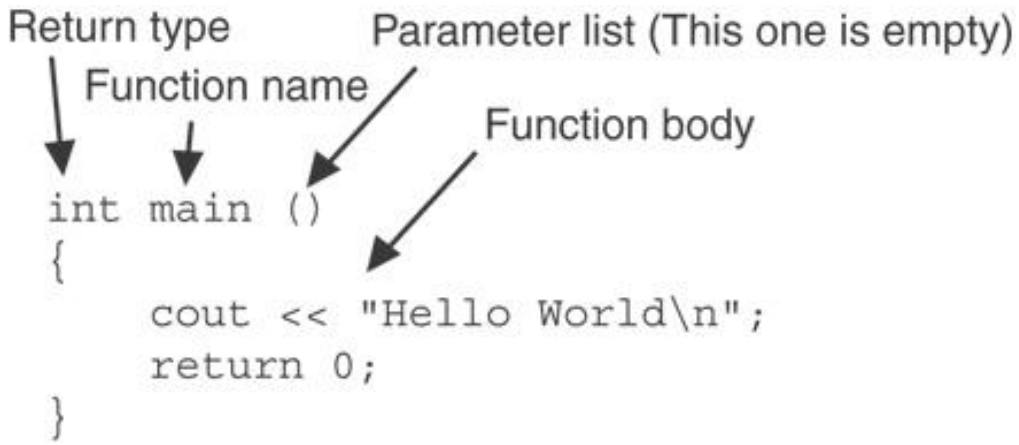
```
#include<iostream>
using namespace std;
int sum(int, int);
int main()
{

    int s;
    s = sum(5, 10);
    cout << "The sum is = " << s;
    return 0;
}

int sum(int a, int b)
{
    int sum = 0;
    sum = a + b;
    return sum;
}
```

Output:
The sum is = 15

Function Definition



Note: The line that reads `int main()` is the *function header*.

Function Return Type

- If a function returns a value, the type of the value must be indicated:

```
int main()
```

- If a function does not return a value, its return type is `void`:

```
void printHeading()
{
      cout << "Monthly Sales\n";
}
```

Calling a Function

- To call a function, use the function name followed by `()` and `;`

```
printHeading();
```

- When called, program executes the body of the called function

- After the function terminates, execution resumes in the calling function at point of call.

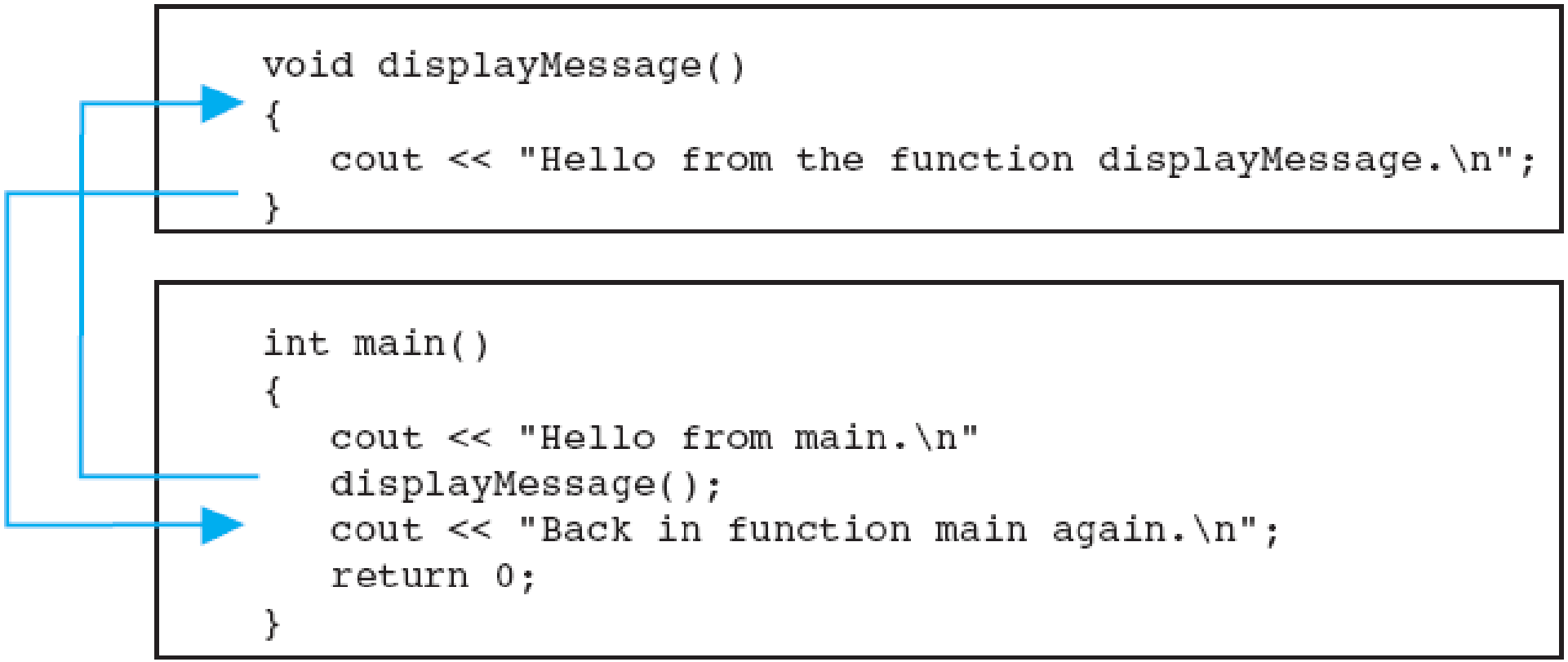
Example Program

```
1  // This program has two functions: main and displayMessage
2  #include <iostream>
3  using namespace std;
4
5  //****************************************
6  // Definition of function displayMessage  *
7  // This function displays a greeting.     *
8  //****************************************
9
10 void displayMessage()
11 {
12    cout << "Hello from the function displayMessage.\n";
13 }
14
15 //****************************************
16 // Function main                          *
17 //****************************************
18
19 int main()
20 {
21    cout << "Hello from main.\n";
22    displayMessage();
23    cout << "Back in function main again.\n";
24    return 0;
25 }
```

Program Output

```
Hello from main.
Hello from the function displayMessage.
Back in function main again.
```

Flow of Control in Program 6-1



```
void displayMessage()
{
   cout << "Hello from the function displayMessage.\n";
}
```

```
int main()
{
   cout << "Hello from main.\n"
   displayMessage();
   cout << "Back in function main again.\n";
   return 0;
}
```

Calling Functions

- `main` can call any number of functions
- Functions can call other functions
- Compiler must know the following about a function before it is called:
  - name
  - return type
  - number of parameters
  - data type of each parameter

Function Prototypes

- Ways to notify the compiler about a function before a call to the function:
  - Place function definition before calling function's definition
  - Use a function prototype (function declaration) – like the function definition without the body
    - Header: `void printHeading()`
    - Prototype: `void printHeading();`

Example program

```
1  // This program has three functions: main, First, and Second.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototypes
6  void first();
7  void second();
8
9  int main()
10 {
11    cout << "I am starting in function main.\n";
12    first();     // Call function first
13    second();    // Call function second
14    cout << "Back in function main again.\n";
15    return 0;
16 }
17
```

(Program Continues)

Example program cont…

```
18  //****************************************
19  // Definition of function first.        *
20  // This function displays a message.    *
21  //****************************************
22
23  void first()
24  {
25     cout << "I am now inside the function first.\n";
26  }
27
28  //****************************************
29  // Definition of function second.       *
30  // This function displays a message.    *
31  //****************************************
32
33  void second()
34  {
35     cout << "I am now inside the function second.\n";
36  }
```

Prototype Notes

- Place prototypes near top of program
- Program must include either prototype or full function definition before any call to the function – compiler error otherwise
- When using prototypes, can place function definitions in any order in source file

Sending Data into a Function

- Can pass values into a function at time of call:

```
c = pow(a, b);
```

- Values passed to function are arguments

- Variables in a function that hold the values passed as arguments are parameters

A Function with a Parameter Variable

```
void displayValue(int num)
{
   cout << "The value is " << num << endl;
}
```

The integer variable `num` is a parameter.
It accepts any integer value passed to the function.

Example program

```
1  // This program demonstrates a function with a parameter.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototype
6  void displayValue(int);
7
8  int main()
9  {
10    cout << "I am passing 5 to displayValue.\n";
11    displayValue(5);  // Call displayValue with argument 5
12    cout << "Now I am back in main.\n";
13    return 0;
14 }
15
```

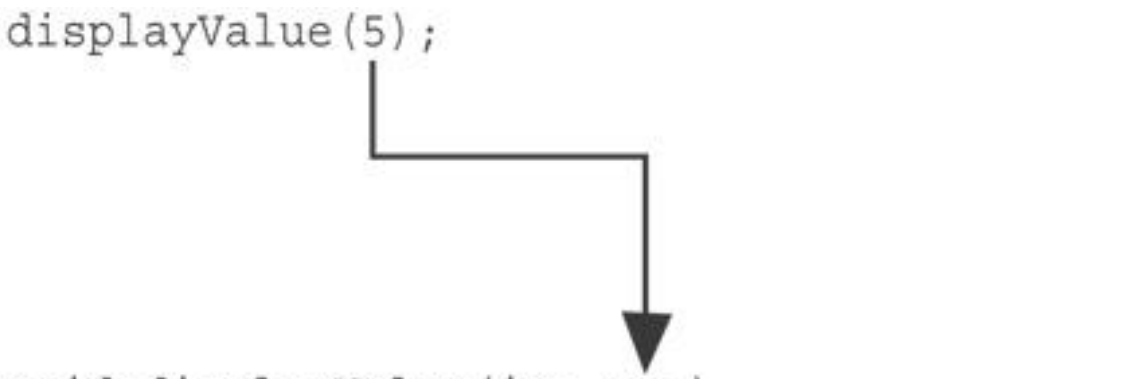
(Program Continues)

Example program cont…

```
16  //*********************************************************
17  // Definition of function displayValue.                   *
18  // It uses an integer parameter whose value is displayed. *
19  //*********************************************************
20
21  void displayValue(int num)
22  {
23     cout << "The value is " << num << endl;
24  }
```

Program Output

```
I am passing 5 to displayValue.
The value is 5
Now I am back in main.
```



The function call in line 11 passes the value 5 as an argument to the function.

Other Parameter Terminology

- A parameter can also be called a formal parameter or a formal argument
- An argument can also be called an actual parameter or an actual argument

Parameters, Prototypes, and Function Headers

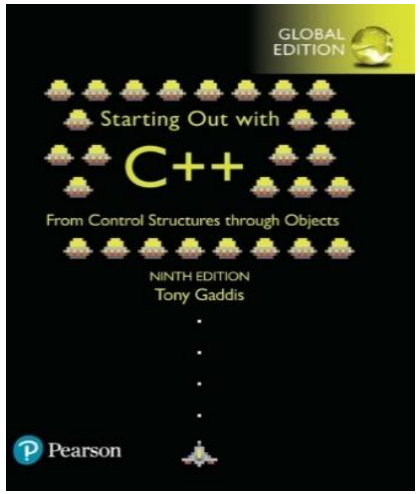
- For each function argument,
  - the prototype must include the data type of each parameter inside its parentheses
  - the header must include a declaration for each parameter in its `()`

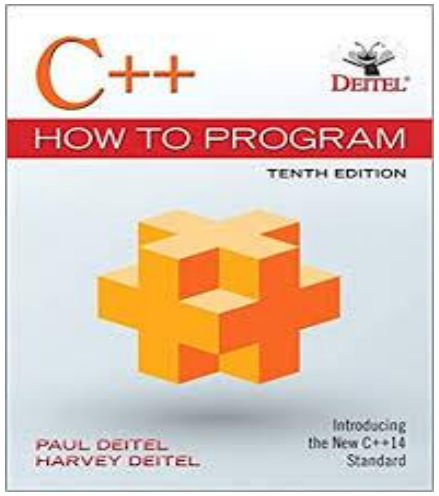
```
void evenOrOdd(int);   //prototype
void evenOrOdd(int num) //header
evenOrOdd(val);         //call
```

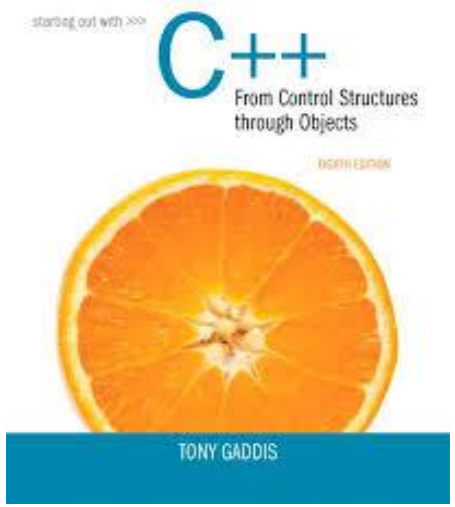
Function Call Notes

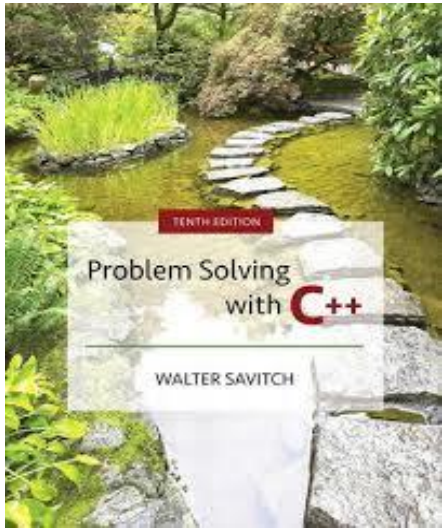
- Value of argument is copied into parameter when the function is called
- A parameter's scope is the function which uses it
- Function can have multiple parameters
- There must be a data type listed in the prototype （）and an argument declaration in the function header （）for each parameter
- Arguments will be promoted/demoted as necessary to match parameters

Thank You All









Acknowledgment: The slides are adapted from the 2012 Pearson Education, Inc.