

# CSCI 152

## Programming Fundamentals II



Summer 2008

---

### User Defined Functions

June 5, 2008

# Objectives

(User Defined Functions Ch 6 & 7)

---

In this lecture/chapter you should review and:

- Learn about user-defined functions
- Examine value-returning functions, including actual and formal parameters
- Explore how to construct and use a value-returning, user-defined function in a program
- Learn how to construct and use void functions



# Functions

---

- Functions are like building blocks
- They allow complicated programs to be divided into manageable pieces
- Some advantages of functions:
  - A programmer can focus on just that part of the program and construct it, debug it, and perfect it
  - Different people can work on different functions simultaneously
  - Can be used in more than one place in a program or in different programs



# User-Defined Functions

---

- Void functions: do not have a data type
- Value-returning functions: have a data type
- To use these functions you need to:
  - Include the correct header file
  - Know the name of the function
  - Know the number of parameters, if any
  - Know the data type of each parameter
  - Know the data type of the value computed by the function, called the type of the function

# Value-Returning Functions



---

- Because the value returned by a value-returning function is unique, we must:
  - Save the value for further calculation
  - Use the value in some calculation
  - Print the value
- A value-returning function is used in an assignment or in an output

# Value-Returning Functions (continued)



---

Properties that form the function definition:

1. Name of the function
2. Number of parameters
3. Data type of each parameter
4. Type of the function
5. Code required to accomplish the task (the body of the function)

# Value-Returning Functions (continued)



---

- Heading: first four properties above
- Formal Parameter: variable declared in the heading
- Actual Parameter: variable or expression listed in a call to a function

# Value-Returning Functions (continued)



---

- The syntax is:

```
functionType functionName(formal parameter  
list)
```

```
{
```

```
    statements
```

```
}
```

- functionType: type of the value returned by the function
  - Also called the data type

# Syntax



---

- The syntax of the formal parameter list is:

dataType identifier, dataType identifier, ...

- The syntax for a function call is:

functionName(actual parameter list)

- The syntax for the actual parameter list is:

expression or variable, expression or

# Functions



---

- The formal parameter list can be empty
- If the formal parameter list is empty
  - Parentheses are still needed
  - Function heading of the value-returning function takes either of the following forms:
    - `functionType functionName()`
    - `functionType functionName(void)`
  - In a function call the actual parameter is empty
- A call to a value-returning function with an empty formal parameter list is:  
`functionName()`



# Value-Returning Functions

---

- To call a value-returning function:
  - Use its name, with the actual parameters (if any) in parentheses
  - There is a one-to-one correspondence between actual and formal parameters

# Value-Returning Functions (continued)



---

- A value-returning function is called in an expression
- Expression may be part of an assignment statement or an output statement
- A function call in a program results in the execution of the body of the called function

# The return Statement



---

- Once the function computes the value, the function returns the value via the return statement
- The syntax of the return statement is:
  - return expression or variable;
- When a return statement executes
  - Function immediately terminates
  - Control goes back to the caller
- When a return statement executes in the function main, the program



# Function Prototype

---

- Function Prototype: function heading without the body of the function
- The syntax is:  

```
functionType functionName(parameter list);
```
- It is not necessary to specify the variable name in the parameter list
- The data type of each parameter must be specified

# Flow of Execution



---

- Execution always begins at
  - The first statement in the function main no matter where main is placed in the program
- Other functions are executed only when they are called

# Flow of Execution (Continued)



---

- Function prototypes appear before any function definition
  - The compiler translates these first
- The compiler can then correctly translate a function call

# Flow of Execution

## (continued)

- A function call statement results in
  - Transfer of control to the first statement in the body of the called function
- After the last statement of the called function is executed
  - Control is passed back to the point immediately following the function call

# Flow of Execution (continued)

- A value-returning function returns a value
- After executing the function
  - The value that the function returns replaces the function call statement



# Summary

---

- Functions (modules) are miniature programs
- Functions enable you to divide a program into manageable tasks
- C++ provides the standard functions
- Two types of user-defined functions: value-returning functions and void functions
- Variables defined in a function heading are called formal parameters
- Expressions, variables, or constant values in a function call are called

# Summary



---

- In a function call, the number of actual parameters and their types must match with the formal parameters in the order given
- To call a function, use its name together with the actual parameter list
- Function heading and the body of the function are called the definition of the function
- If a function has no parameters, you need empty parentheses in heading and call
- A value-returning function returns its



# Summary

---

- A prototype is the function heading without the body of the function; prototypes end with the semicolon
- Prototypes are placed before every function definition, including main
- User-defined functions execute only when they are called
- In a call statement, specify only the actual parameters, not their data types