

Programmer-Defined Functions

Lecture 15

Sections 6.1 - 6.4

Robb T. Koether

Hampden-Sydney College

Fri, Oct 4, 2013

- 1 The `cctype` Library
- 2 Programmer-Defined Functions
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files
- 5 Assignment

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files
- 5 Assignment

The `cctype` Library

`cctype` Functions

bool isalpha(char c);	bool isblank(char c);
bool islower(char c);	bool iscntrl(char c);
bool isupper(char c);	bool isgraph(char c);
bool isdigit(char c);	bool isprint(char c);
bool isxdigit(char c);	bool isspace(char c);
bool isalnum(char c);	void tolower(char c);
bool ispunct(char c);	void toupper(char c);

- `isxdigit()` – Returns **true** if the character is a hexadecimal digit.
- `iscntrl()` – Returns **true** if the character is a control character.

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions**
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files
- 5 Assignment

Programmer-Defined Functions

- A programmer may create his own functions.
- If his function is invoked in `main()`, then
 - Execution leaves `main()` (which is itself a function) and goes to the function.
 - The function executes as a program in its own right.
 - When the function is finished, execution returns to `main()` and continues.

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions**
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files
- 5 Assignment

The Form of a Function

The Form of a Function

```
return-type function-name(formal-param-list)  
{  
    function-body  
    return return-value;  
}
```

- *function-name* is the name that the programmer chooses for the function.
- The *formal-param-list* is a list of object types and names passed to the function as parameters.
- The *function-body* is a sequence of C++ statements that will compute the appropriate value for the returned object.

The Form of a Function

The Form of a Function

```
return-type function-name(formal-param-list)  
{  
    function-body  
    return return-value;  
}
```

- *return-type* is the type of object that the function returns to the **calling function**.
- *return-value* must be of type *return-type*.
- *return-type* may be **void**, in which case the **return** statement is optional.
 - If it is written, then there is no return value specified.

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions**
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files
- 5 Assignment

Example of a Function

Function Definition

```
float average3(float a, float b, float c)
{
    float avg = (a + b + c)/3.0;
    return avg;
}
```

- This function will return the average of three numbers.

Example of a Function

Function Definition

```
float average3(float a, float b, float c)
{
    return (a + b + c)/3.0;
}
```

- It is legal to return the value of an expression.

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions
 - The Form
 - Examples
- 3 Function Calls**
- 4 Function Prototypes and Header Files
- 5 Assignment

Function Calls

Function Call

function-name(actual-param-list)

- The *actual-param-list* is a list of actual objects or expressions.
- A **function call** occurs in an expression.
- If the function is **void** type, then the function call is a complete statement and must occur on a line by itself.
- If the function is not **void** type, then the function call occurs within an expression where an object of the function type is permitted.

Function Usage

- Data types are not specified in the actual parameter list.
- Actual vs. formal parameters
 - Types must match, or else
 - There must be a conversion rule to convert the actual type into the formal type.

Example of Function Usage

Function Usage

```
cout << "Enter the three grades: ";  
float grade1, grade2, grade3;  
cin >> grade1 >> grade2 >> grade3;  
float avg_grade = average3(grade1, grade2, grade3)  
cout << avg_grade << endl;
```

- This program finds the average of three grades.

Example of Function Usage

Function Usage

```
cout << "Enter the three grades: ";  
float grade1, grade2, grade3;  
cin >> grade1 >> grade2 >> grade3;  
cout << average3(grade1, grade2, grade3) << endl;
```

- The function call may be placed in the output statement.

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files**
- 5 Assignment

Function Prototypes

- The function prototype must precede the function definition and usage.
- Place the prototype at the top of the implementation file (`.cpp`) in which the function is used.
- Or, place the prototype in a header (`.h`) file and include the header file in the implementation file.

Example of a Prototype

Function Prototype

```
float average(float a, float b, float c);
```

```
int main()
```

```
{
```

```
    ⋮
```

```
    average(x, y, z)
```

```
    ⋮
```

```
}
```

```
float average(float a, float b, float c)
```

```
{...}
```

- The prototype of the `average()` function.

Function Execution

- The values of the objects in the *actual-parameter-list* are passed to the corresponding objects in the *formal-parameter-list*.
- The *function-body* is executed.
- The returned value is substituted into the expression in place of the function reference in the calling function.

Examples of Programmer-Defined Functions

- Examples

- `AverageFunc.cpp`
- `IsVowelFunc.cpp`
- `IsVowelFunc2.cpp`, `isvowel.cpp`, `isvowel.h`

Outline

- 1 The `cctype` Library
- 2 Programmer-Defined Functions
 - The Form
 - Examples
- 3 Function Calls
- 4 Function Prototypes and Header Files
- 5 Assignment

Assignment

Assignment

- Read Sections 6.1 - 6.4.