Programmer-Defined Functions

Lecture 16
Sections 6.1 - 6.4

Robb T. Koether

Hampden-Sydney College

Wed, Oct 3, 2018

Outline

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);     char tolower(char c);
bool ispunct(char c);     char toupper(char c);
```

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

Outline

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 beginning of the function.
  - The function executes its instructions program from top to bottom, or until it hits a **`return`** statement.
  - When the function is finished, execution returns to its departure point in `main()` and continues.

Outline

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 (i.e., declarations) 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

Example of a Function

Function Definition

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

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

Example of a Function

Function Definition

```
bool isOdd(int n)
{
    return n % 2 == 1;
}
```

- This function will return the average of three numbers.

Outline

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

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 average3(float a, float b, float c);

int main()
{
      ⋮
    average3(x, y, z)
      ⋮
}

float average3(float a, float b, float c)
{...}
```

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

Examples of Programmer-Defined Functions

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

Outline

Assignment

Assignment

- Read Sections 6.1 - 6.4.