C++ Programming

Ch. 2 Setting Out to C++

Spring 2014

Myung-Il Roh

Department of Naval Architecture and Ocean Engineering
Seoul National University

Ch. 2 Setting Out to C++

Contents

- ☑ Review
- ☑ Structure of the C++ Program
- ☑ The C++ Preprocessor and the 'iostream' File
- ☑ Input and Output Statements
- ☑ Functions
- ☑ C++ Coding Rules: Identifiers & Keywords
- ☑ Summary
- ☑ Practice

Review

- ☑ Programming Using Microsoft Visual C++
  - Create a Project, Coding, Compile, Link, and Execute
  - “Hello World!” Program

Structure of C++ Program (1/3)

☑ Example of C++ Program

```
#include <iostream>
using namespace std;
```
(1)

```
double stone2kg(double);
```
(2)

```
double main()
{
        double stone, kgs;
        cout << “input weight in unit stone”;
        cin >> stone;
        kgs = stone2kg(stone);
        cout << stone << “ Stone is ”;
        cout << kgs << “kilos.\n”;
        return 0;
}
```
(3)

Definition of a main function

```
double stone2kg(double sts)
{
        return 6.35 * sts;
}
```

Definition of a user-defined function

Structure of C++ Program (2/3)

(e.g., #include)

- A preprocessor is a program that processes a source file before the main compilation takes place.
- A directive is a typical preprocessor action: adding or replacing text in the source code before it's compiled.
- Ex. #include <iostream>
  - This directive causes the preprocessor to add the contents of the 'iostream' file to your program.
- The preprocessor directive must .

to be used in the source code.

Structure of C++ Program (3/3)

- The ‘main()’ function must be exist .

C++ Preprocessor and 'iostream' File

☑ #include <iosream>

- A statement starting with '#' is called ' '.
  - Example of preprocessors: file inclusion(#include), macro expansion(#define), and conditional compilation(#if, #ifdef).
  - 'iostream': Defines input and output functions in C++.
    - When using 'cin' and 'cout' for input and output, the 'iostream' file must be included.
    - In C language '.h' is used as extension for header files, but in C++ language, the extension can be omitted. For this, the statement of 'using namespace std;' is required.

- Format of the preprocessor directive
  - #include <aaaaa> // For header files in C++ standard library
  - #include <aaaaa.h> // For header files in C standard library
  - #include <myheader.h> // For user-defined header files

Input and Output Statements

(using ‘ ’ object)

- ‘cout’ judges the data type of a variable automatically.
- ‘cout’ can use continuously
  - *cout << “Wow, ” << 8 << “ dogs!”*
    - ❖ Result: *Wow, 8 dogs!*
- In C language, the ‘ ’ function is used for output.

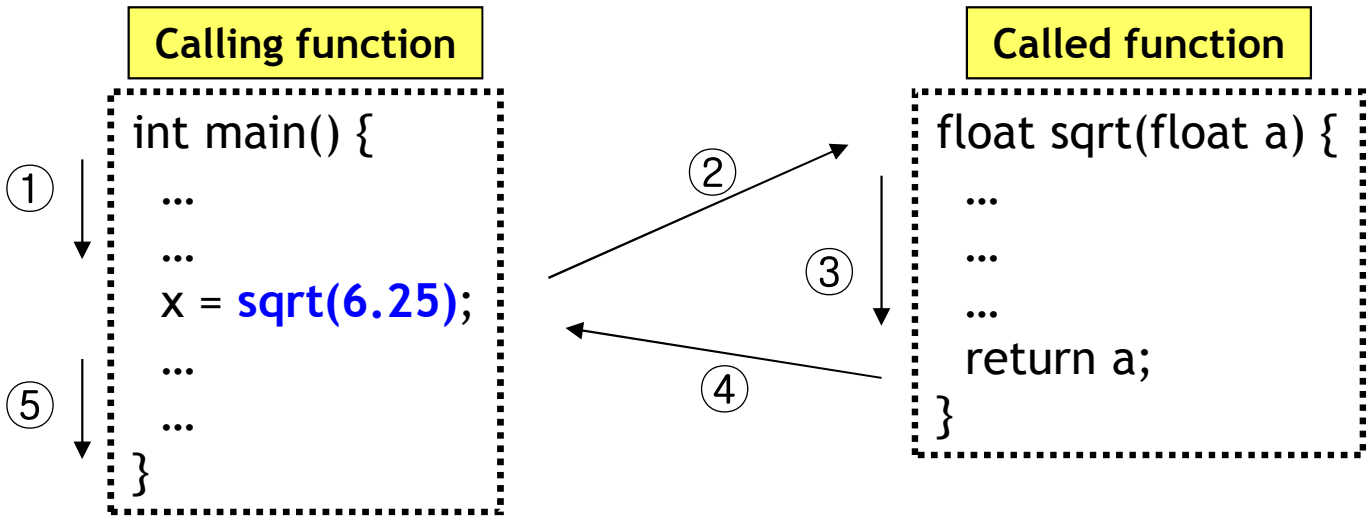
(using ‘ ’ object)

- ‘cin’ stores the input data on a designated variable, and judges the data type of the variable automatically.
  - *cin >> score;* // For C user: Warning! it is not ‘&score’.
- In C language, the ‘ ’ function is used for input.

Functions (1/5)

☑ Function

- **Standard library functions vs. User-defined functions**
  - **Standard library functions: Basic functions given in C++ language**
  - **User-defined functions: Functions which are provided by the user.**
- **Depend on the return value**
  - **Function having a return value**
    - » **Ex. The statement like "y = 3,0 * sqrt(2,0);" can be used in the source code.**
  - **Function having no return value**
    - » **Ex. void type function**



Functions (2/5)

Preprocessor directive

```
#include <iostream>
using namespace std;
```

// This program doesn't have a function prototype and declaration of global variables.

Function definition

```
int main( )                                          // Function header
{                                                    // beginning of function body
   cout << "print a message through the monitor \n"; // print
   return 0;                                         // end of main()
}
```

New line

Functions (3/5)

☑ Function Header

.

- Ex. *int main()*

  ('int'): The data type of a value that the function returns

  or (inside of '( )'): The list of parameters to be passed to the function when the function is called, including the data type, order, number of the parameters, etc.

- The 'main()' function must be exist one time in the program because the function is a special one which have to be run at first.

☑ Function Body

- It represents

  .

- The portion enclosed in braces ({ and })

Functions (4/5)

☑ Using Functions

- A C++ program should provide a prototype for each function used in the program.
- Format of a function prototype
  - *double swap(double, int);*
    - Function name: swap
    - Parameters or arguments: double and int
    - Data type of return value: double

- Declaration position of the function prototype
  - Declare the function prototype before the main() function in the source code.
  - or, declare it in the user-defined header file and then include the header file in the source code.

.

Functions (5/5)

☑ User-defined functions

```
#include <iostream>
using namespace std;
```

(1) Preprocessor directive

```
double stone2kg(double);
```

(2) Function prototype and declaration of global variables

```
double main()
{
        double stone, kgs;
        cout << “input weight in unit stone”;
        cin >> stone;
        kgs = stone2kg(stone);
        cout << stone << “ Stone is ”;
        cout << kgs << “kilos.\n”;
        return 0;
}
```

(3) Function definition (header + body)

Definition of a main function

```
double stone2kg(double sts)
{
        return 6.35 * sts;
}
```

Definition of a user-defined function

C++ Coding Rules - Identifiers & Keywords (1/2)

☑ Coding Rules and the Comment

- The comment can be on its own line or it can be on the same line as code.
  - *// This is line for comment.*
  - *int a; // Comment for the variable*
- The comment runs from the '//' to the end of the line.
- In C language '/*' (start of comment) and '*/' (end of comment) are used.

☑ Using Character

- Uppercase and lowercase alphabet
- Case sensitive: discriminates between uppercase and lowercase.
- Number: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (digit)
- Special character: + - / = ( ) { } < > ' [ ] ! # $ % & _ | ^ ~ ₩ . ; : ?

C++ Coding Rules - Identifiers & Keywords (2/2)

- User-defined identifier: Variables, function name, array name, etc. except for keywords in C++ language
  - Identifier: Variables, function name, array name, etc. that the user define. (e.g., Korea, ROK, …)
  - The first letter of the identifier should be start with the alphabet or '_' (underscore).
  - There is no limitation in the length for the identifier.

- Reserved words which are used by the compiler
- Identifiers used by the compiler
- Ex. break, if, else, long, switch, case, enum, register, typedef, char, …

Summary (1/3)

☑ A C++ program consists of .

☑ Programs begin executing at the beginning of the function called 'main()' (all lowercase), so you should always have a function by this name.

.

- The function header tells you what kind of return value, if any, the function produces and what sort of information it expects arguments to pass to it.
- The function body consists of a series of C++ statements enclosed in paired braces ('{}').

Summary (2/3)

☑ C++ statement types include the following:

: It announces the name and the type of a variable used in a function.

: It uses the assignment operator ('=') to assign a value to a variable.

: It sends a message to an object, initiating some sort of action.

: It activates a function. When the called function terminates, the program returns to the statement in the calling function immediately following the function call.

: It declares the return type for a function, along with the number and type of arguments the function expects.

: It sends a value from a called function back to the calling function.

Summary (3/3)

- ☑ C++ provides two predefined objects ('cin' and 'cout') for handling input and output.

- ☑ C++ can use the extensive set of C library functions. To use a library function, that provides the prototype for the function.

Practice

☑ **Make a program that gets 2 variables, adds variables, and shows its result.**

```
Preprocessor directives

int main()
{
    int x, y, z;  // declare integer variables x, y, and z.
    input value to variable x.
    input value to variable y.
    z = x + y;
    print the value of z.
    return 0;
}
```