

# Programming Languages for MIS

## Chapter 2. C++

# Prerequisite 1

## 1. Variable

A variable or data item is the name of a piece of CPU memory that holds data. The data is called the value of the variable/data item. The original value of a variable could be a default value (such as 0). The value of a variable can be changed through operations, but can never be lost.

# Prerequisite 2

## 2. Arithmetic operation

Arithmetic operations in procedural programming are similar to day-to-day arithmetic calculation, but use reverse expression. For instance, instead of  $a+b=c$ ,  $c=a+b$  is used in programming, which means: let  $c$  equal to  $a$  plus  $b$ .

# Prerequisite 3

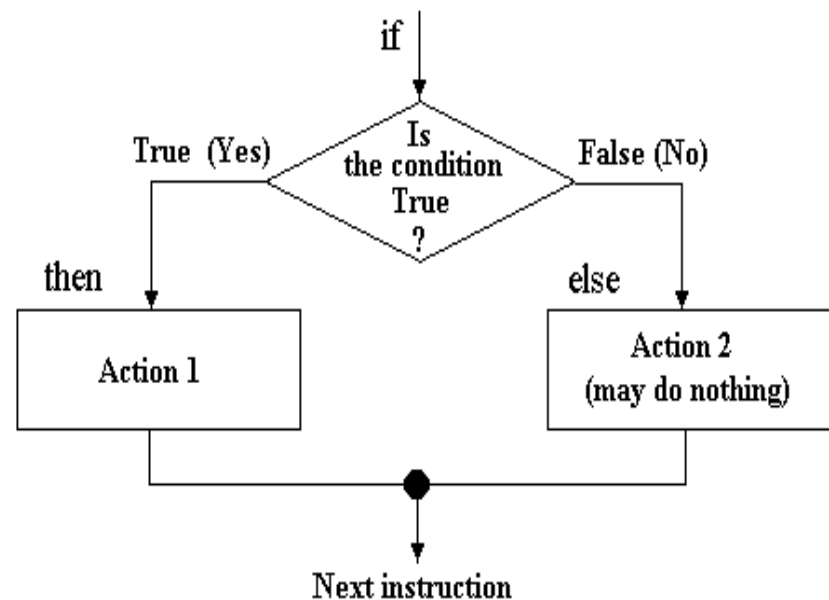
## **3. Execution sequence**

During the execution of the procedure of a program, instructions are executed one after another in a sequence (so called execution sequence) in which they are encountered, but not in the order they are listed in the program. Statements of if-then and loops can control the execution sequence of the program.

# Prerequisite 4

## 4. If-then-else logic

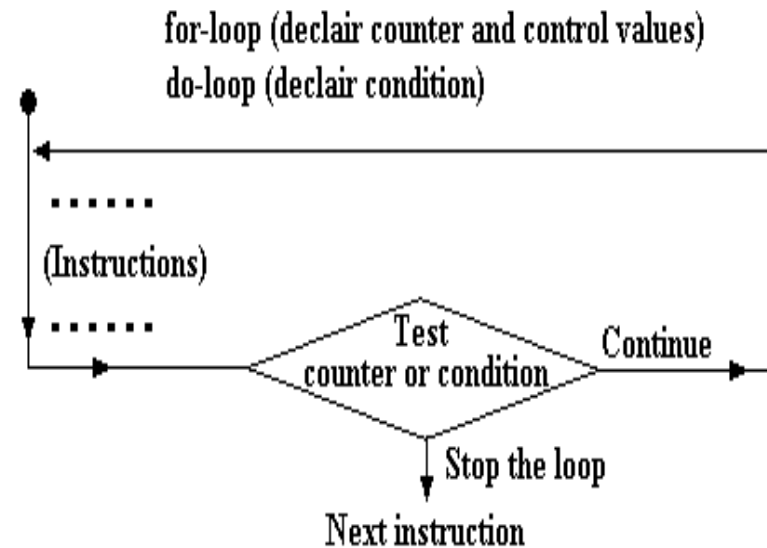
An if-then-else statement controls the computer execution sequence based on a current condition. The logic is illustrated in figure.



# Prerequisite 5

## 5. Loop

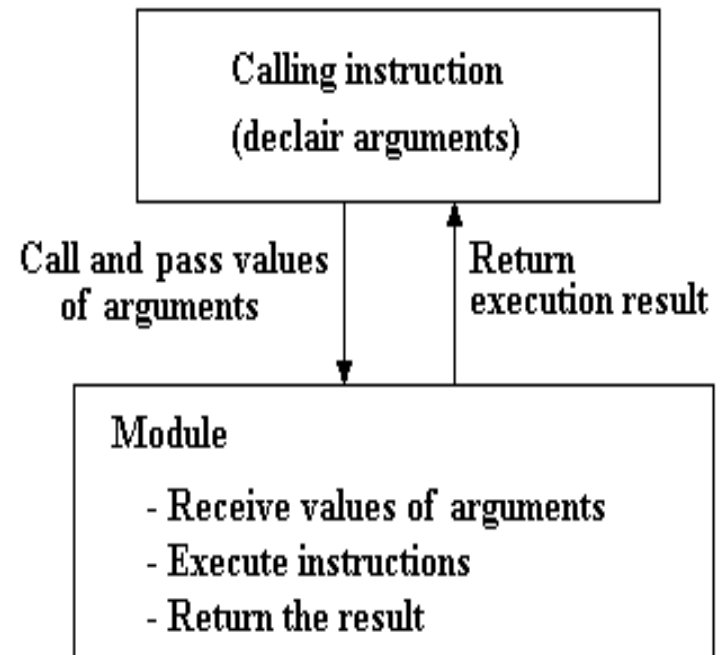
A loop is a group of instructions which is specified once but are carried out several times in succession. A loop statement defines such an iteration procedure, as illustrated in the figure. The common loops include for-loop and do-loop



# Prerequisite 6

## 6. Call module

A large program must be divided into modules for better quality and re-use. Here, a module could be a paragraph of instructions, a function, a method, a subroutine, depending upon a specific language. An instruction can call a module to accomplish a specific task carried out by the module, as illustrated in the figure.



# C and C++

- C++ is an extension of C
- Strictly speaking, C is structured programming language, C++ is object-oriented programming language. Now C++ is a superset of C and C++.
- They share similar syntaxes
- Their features are followed by many other languages, such as JavaScript, PHP, C#



# A Simple C Program

```
/*    C Programming Example    */  
  
#include<iostream>  
using namespace std;  
void main()  
{  
    printf("Hello, world ! \n");  
}
```

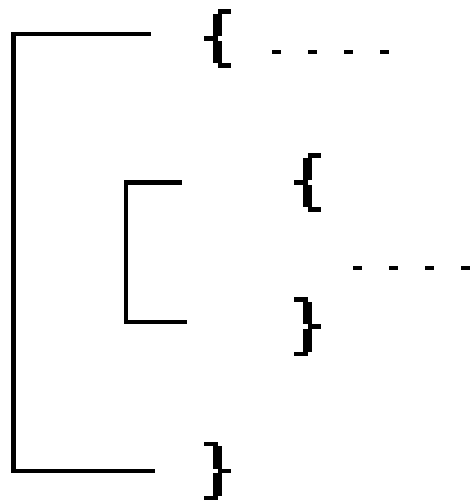
# Your turn:

## Hands-on of “Hello, World!”

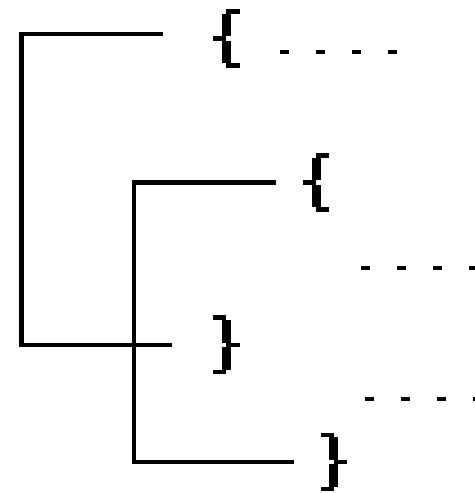
To run the C program in Microsoft Visual Studio

- Edit – Edit source code
- Compile – Compile the program
- Run – Run the executable code

# Structure of Programs



**Correct**



**Never happen**

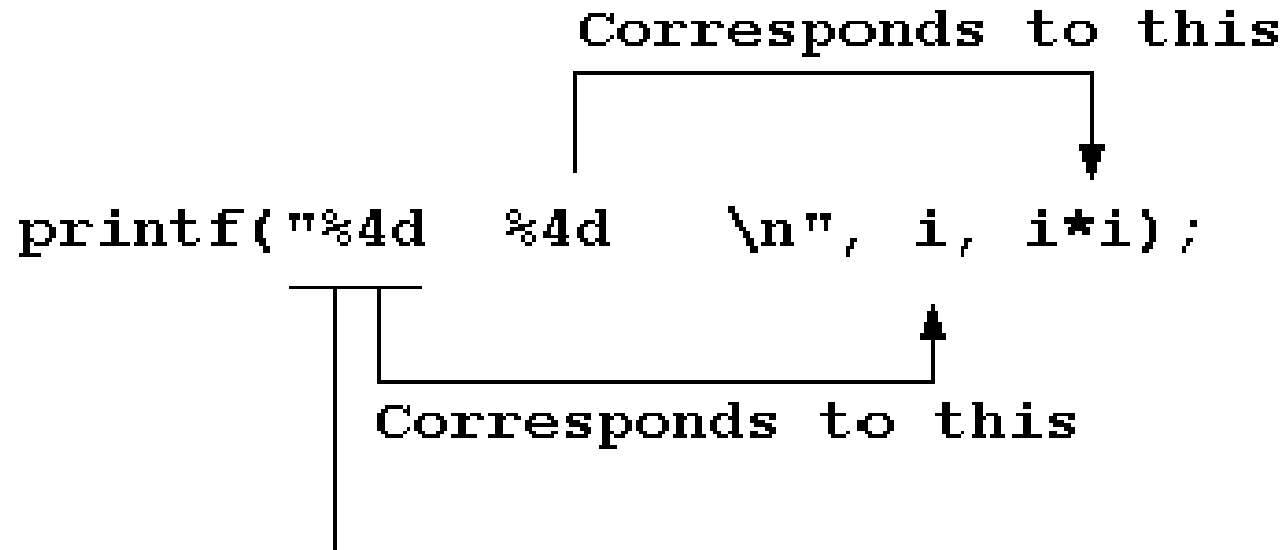
# Go through Listing 2.2

- Go through the C program line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - Format
  - Key words
  - User-defined words
  - Syntax of statement

# for loop

```
for(int [counter]=1;  
    [counter]<=[final value of the counter];  
    [counter]++)  
  
    { [repetition actions] };
```

# printf and Conversion Specifier



# Arithmetic operations

$x = a + b$

$x = a - b$

$x = a * b$

$x = a / b$

$x = \text{pow}(a, b)$

$y = x++$

$y = x--$

# if statement

```
if ( [condition] )  
    { [action_1] ; }  
else  
    { [action_2] ; } ;
```



# Go through Listing 2.3

- Go through the C program line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - Format
  - Key words
  - User-defined words
  - Syntax of statement

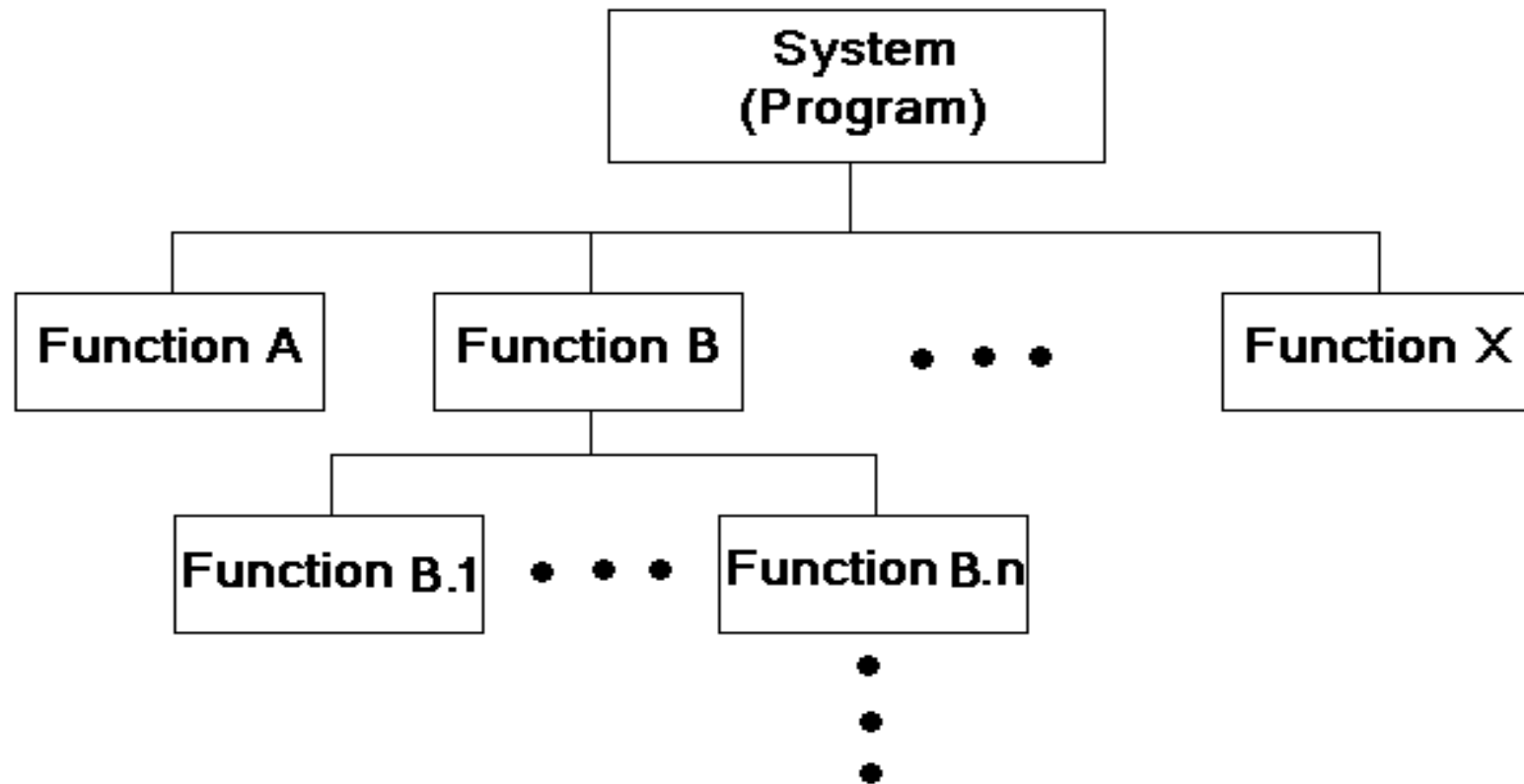
# string

```
char mystring[16];
```

```
strcpy([destination string],  
       [source string]);
```

```
strcmp([string-1], [string-2]);
```

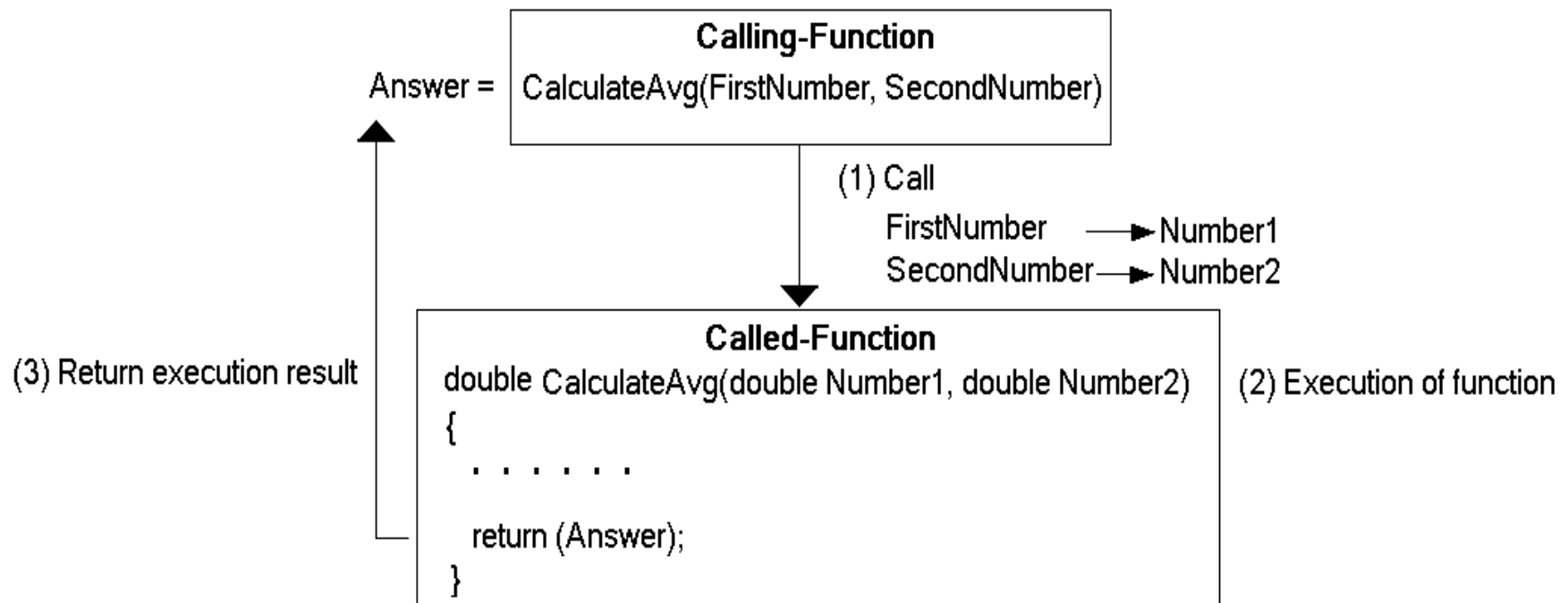
# Function Oriented Programming



# Go through Listing 2.4

- Go through the C program line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - Format
  - Key words
  - User-defined words
  - Syntax of statement

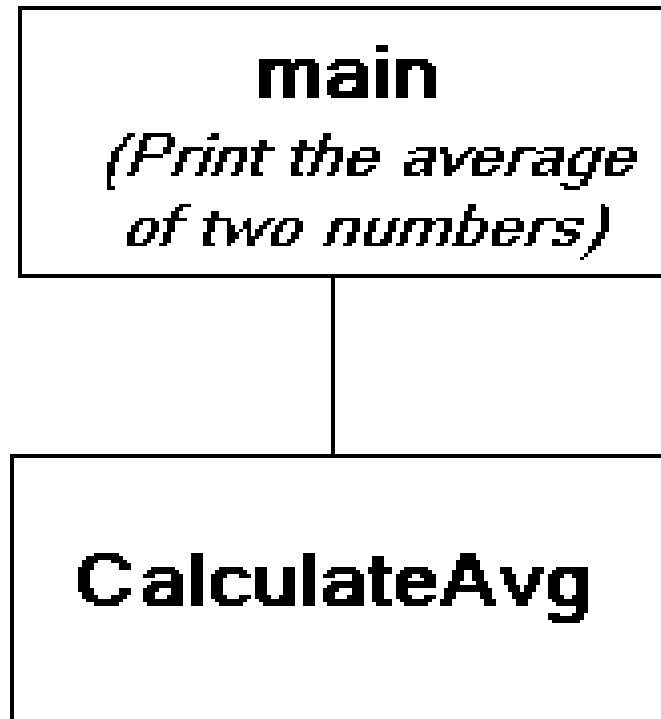
# Calling Function and Called Function



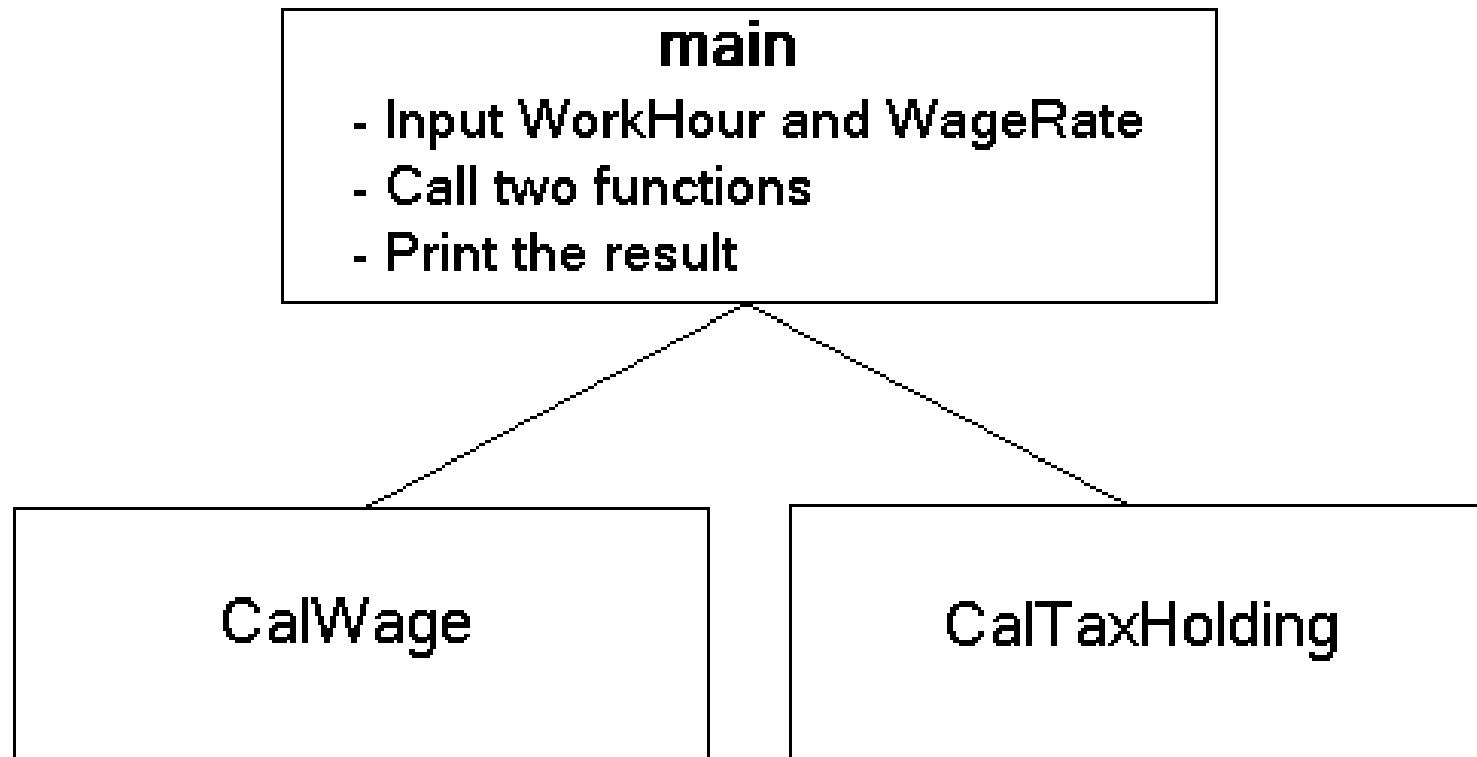
# User Defined Function

[function type] [*function name*] (*data types of arguments*);

# Structure Diagram



# Two functions

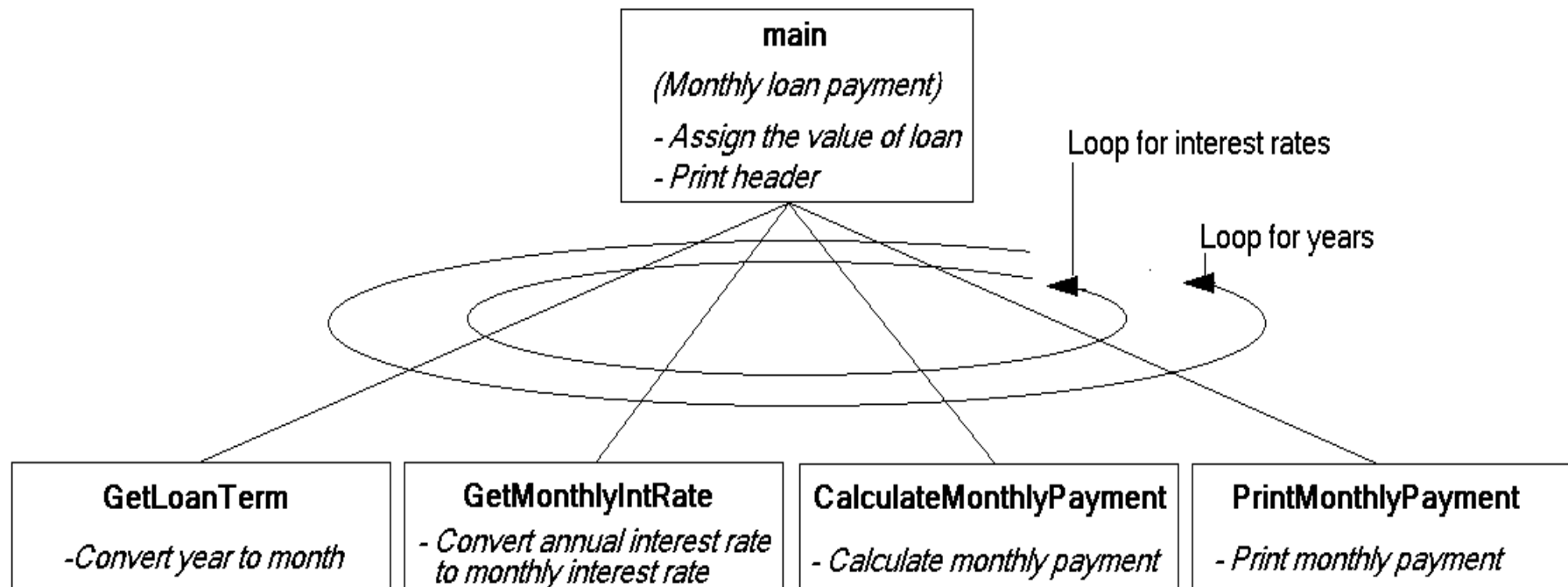




# Go through Listing 2.5

- Go through the C program line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - Format
  - Key words
  - User-defined words
  - Syntax of statement

# Structured Diagram of Listing 2.6



# Go through Listing 2.6

- Go through the C program line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - Format
  - Key words
  - User-defined words
  - Syntax of statement

# Object-Oriented Programming

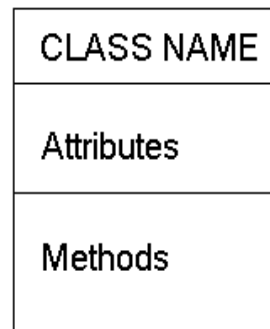
- **Object:**

- customer – physically existing body;
- inventory item – physically existing good;
- ticket – a document;
- game – an event;
- button on screen – a widget representing a program.

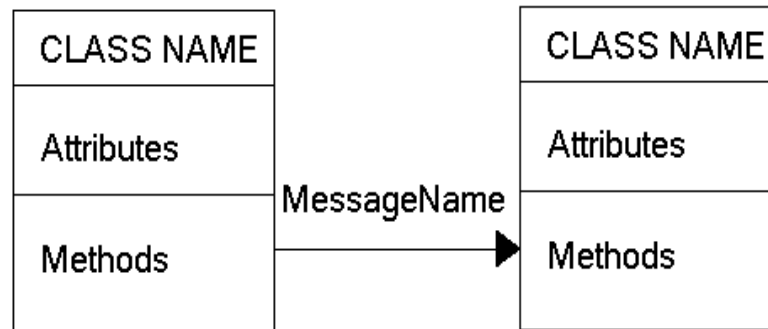
# OOP (cont'd)

- **Class**
  - CUSTOMER – class of customers;
  - INVENTORY – class of inventory items;
  - TICKET – class of tickets;
  - GAME – class of games;
  - BUTTON – class of buttons.
- **OOP is actually “Class-oriented programming”**

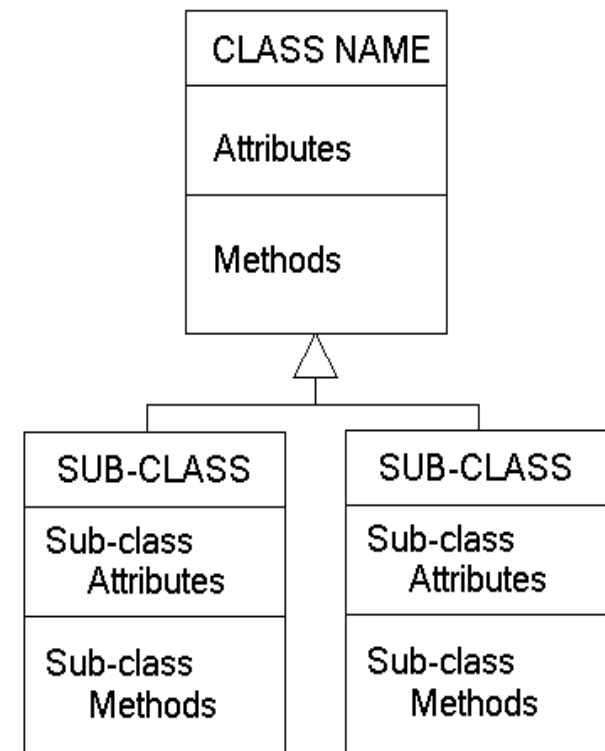
# OOP Constructs



**Class**



**Message Sending**



**Inheritance**

# Class

```
class class_name_identifier
{
private:
    data and methods
    // The data and methods cannot be accessed
    directly
public:
    data and methods
    // The data and methods can be accessed directly
};
```

# Go through Listings 2.8 and 2.9

- Go through the C++ program line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - The relationships between the two programs
  - Syntax of statement



# Terminology

- `public` and `private`
- Constructor
- Method

# Declare objects

```
class_name    object_name;
```

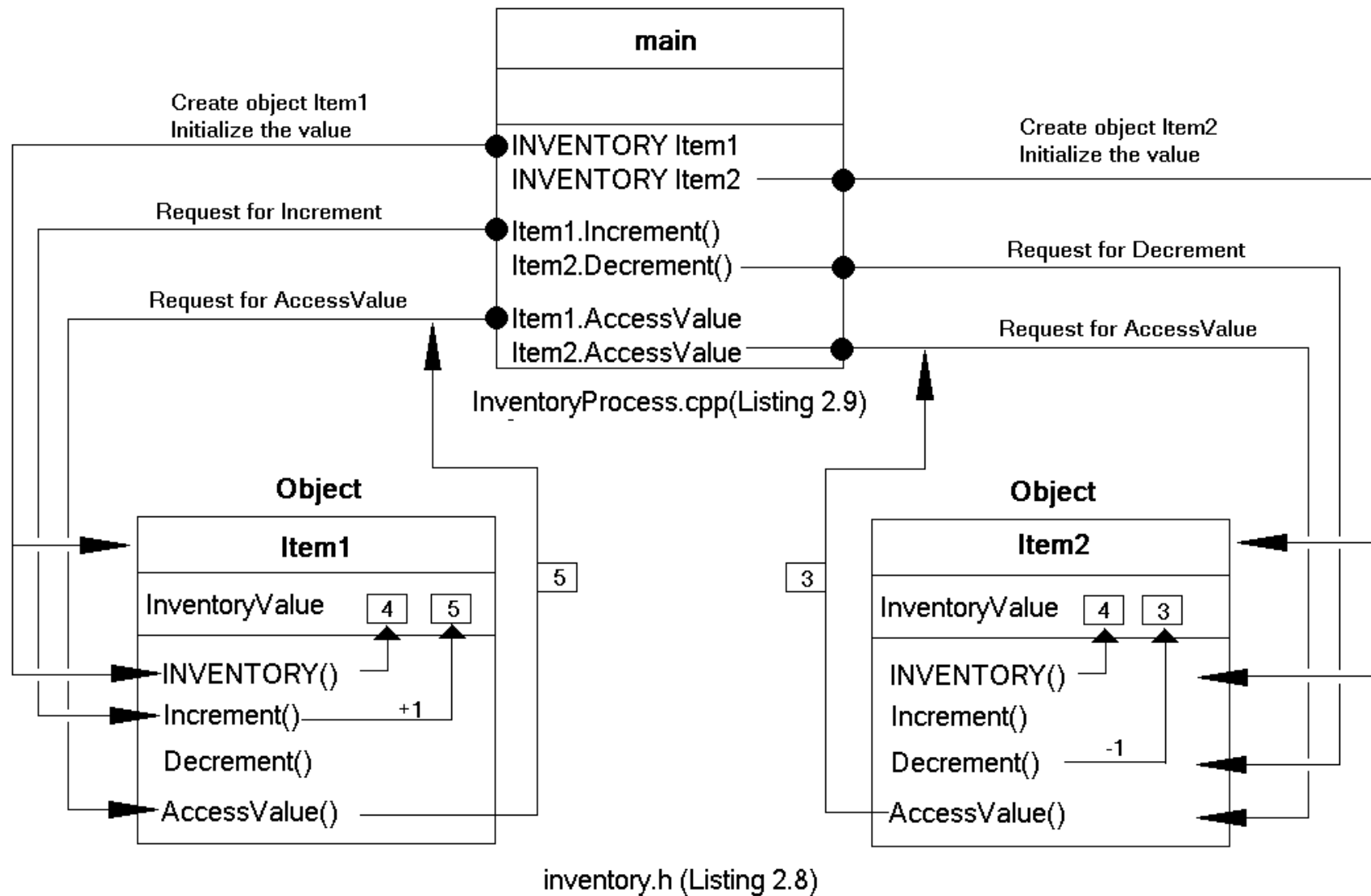
The computer creates an object in this class.

# Message Sending

```
object_name.method_name;
```

The concept of message sending in OOP is not much different from calling function in structured programming

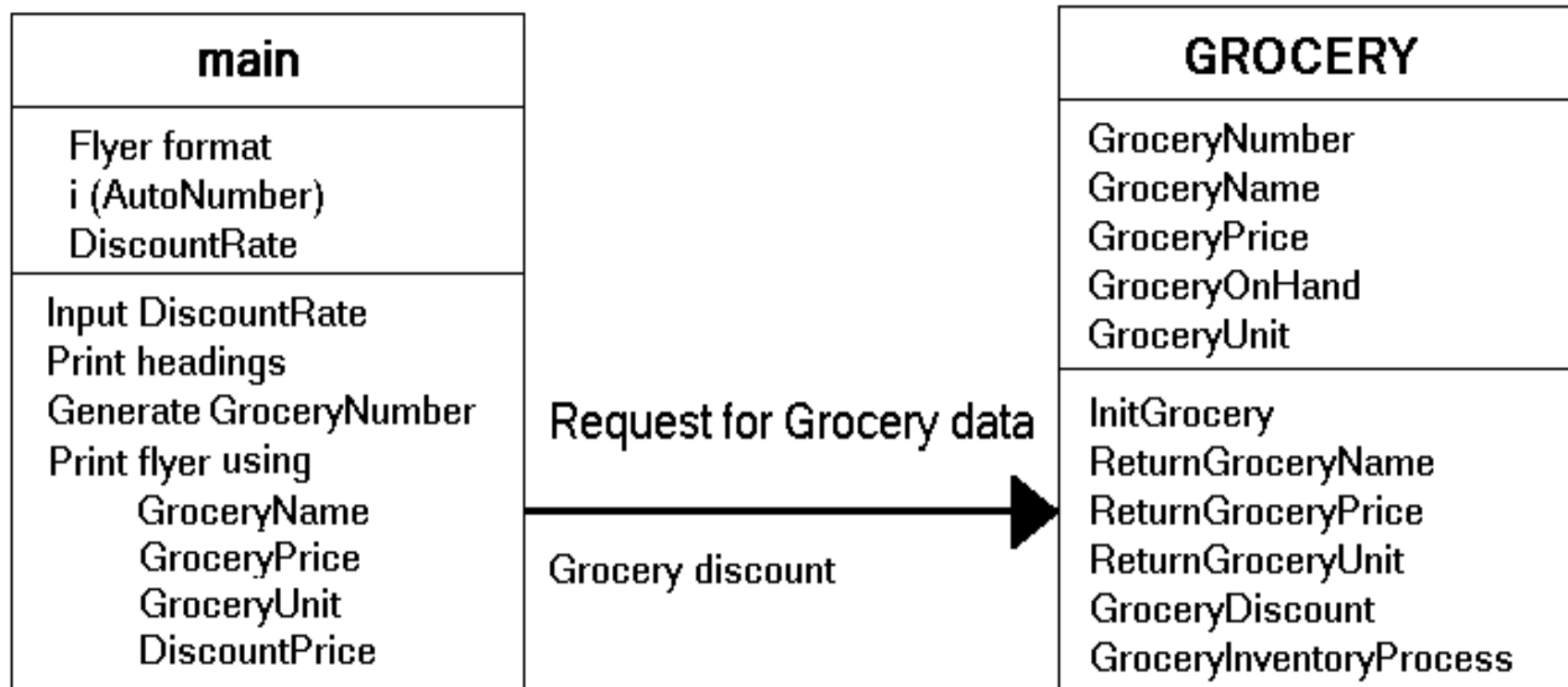
# Walkthrough of Listings 2.8 and 2.9



# Go through Listings 2.10(a) and 2.10(b)

- Go through the C++ programs line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - The relationships between the two programs
  - Syntax of statement

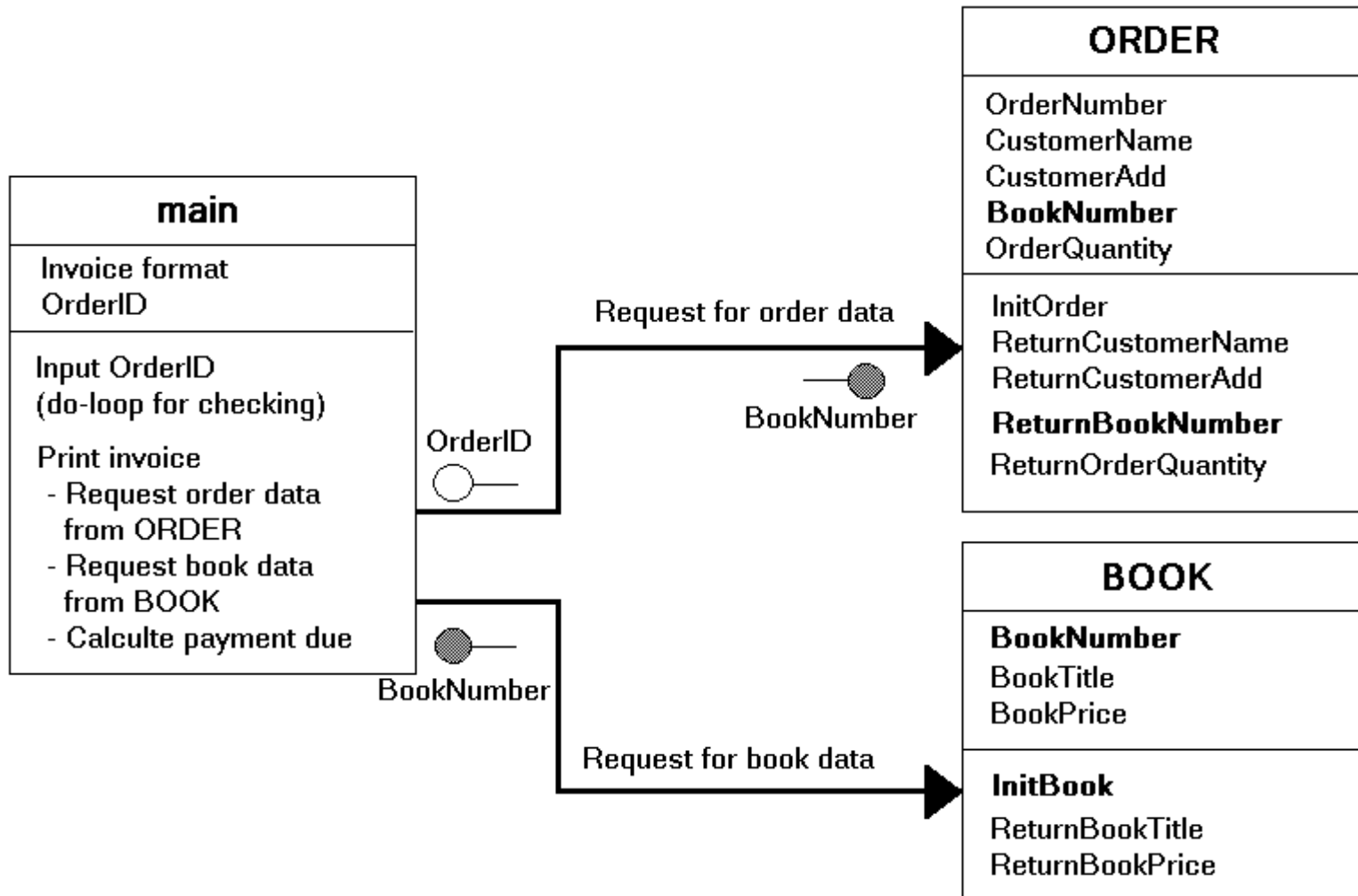
# OOP Diagram of Listings 2.10(a) and 2.10(b)



# Go through Listings 2.11(a), 2.11(b), and 2.11(c)

- Go through the C++ programs line-by-line to understand each statement.
- The textbook describes it in detail
- Note the important points
  - The relationships between the three programs
  - Syntax of statement

# OOP Diagrams for Listings 2.11(a), 2.11(b), and 2.11(c)





## OOP Diagram for Listing 2.12(a) and Listing 2.12(b)

