XIAMEN UNIVERSITY MALAYSIA
厦門大學马来西亚分校

CST209

Object-oriented Programming C++

(Week 4)

Dr. Teo Bee Guan

Content

- Function
- Struct
- Object-oriented programming
    - Introduction to class and object

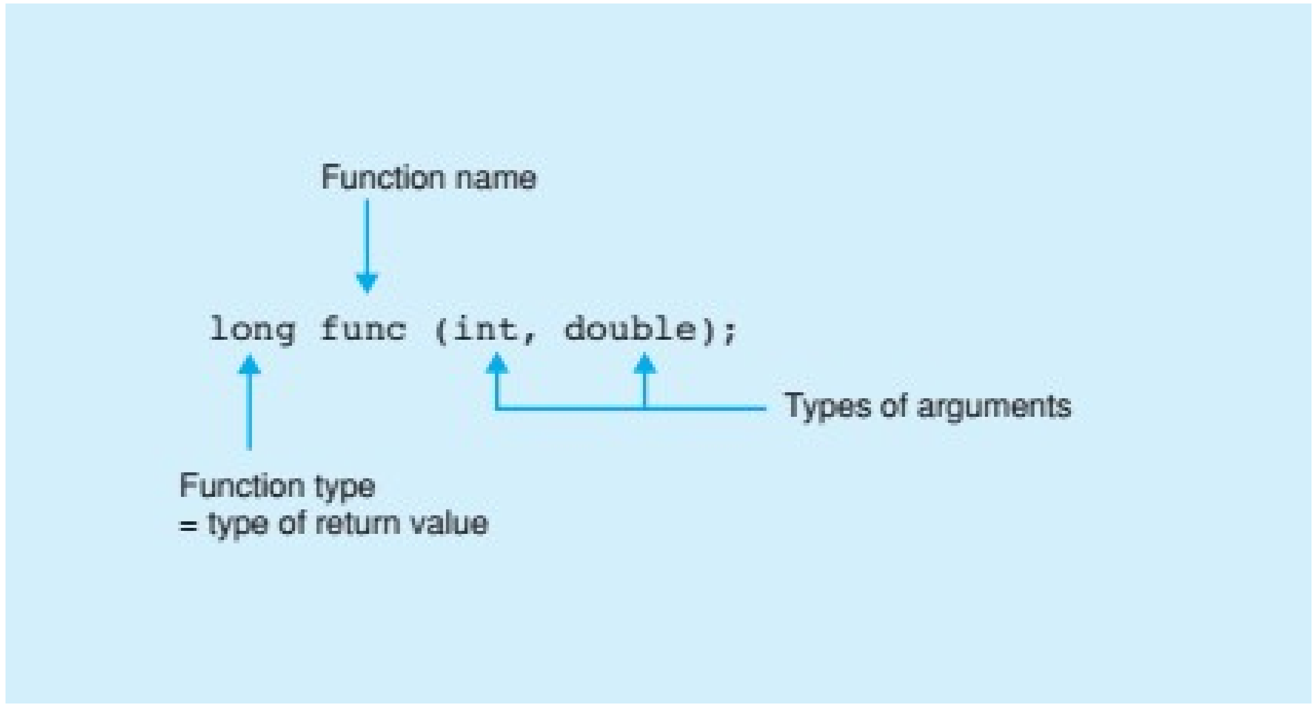
Function

- A function is a collection of statements that performs a specific task.
- A program may be broken up into manageable functions.
- Functions are commonly used to break a problem down into small manageable pieces.
- Instead of writing one long function that contains all of the statements necessary to solve a problem, several small functions that each solve a specific part of the problem can be written.

Function declaration

The prototype yields the following information to the compiler:

- func is the function name
- The function is called with two arguments: the first argument is of type int, the second of type double
- The return value of the function is of type long.



Function declaration

A function has a name and a type, much like a variable. The function's type is defined by its return value. In addition, the type of arguments required by a function is important. When a function is declared, the compiler must therefore be provided with information on:

1. The name and type of the function
2. The type of each argument.

Examples:

- int sum (int,int);
- bool isEven (int);

Practice: Example 1, 2

void Function

- It isn’t necessary for all functions to return a value.

- Some functions simply perform one or more statements, which follows terminate. These are called **void functions**.

Practice: Example 3

Struct

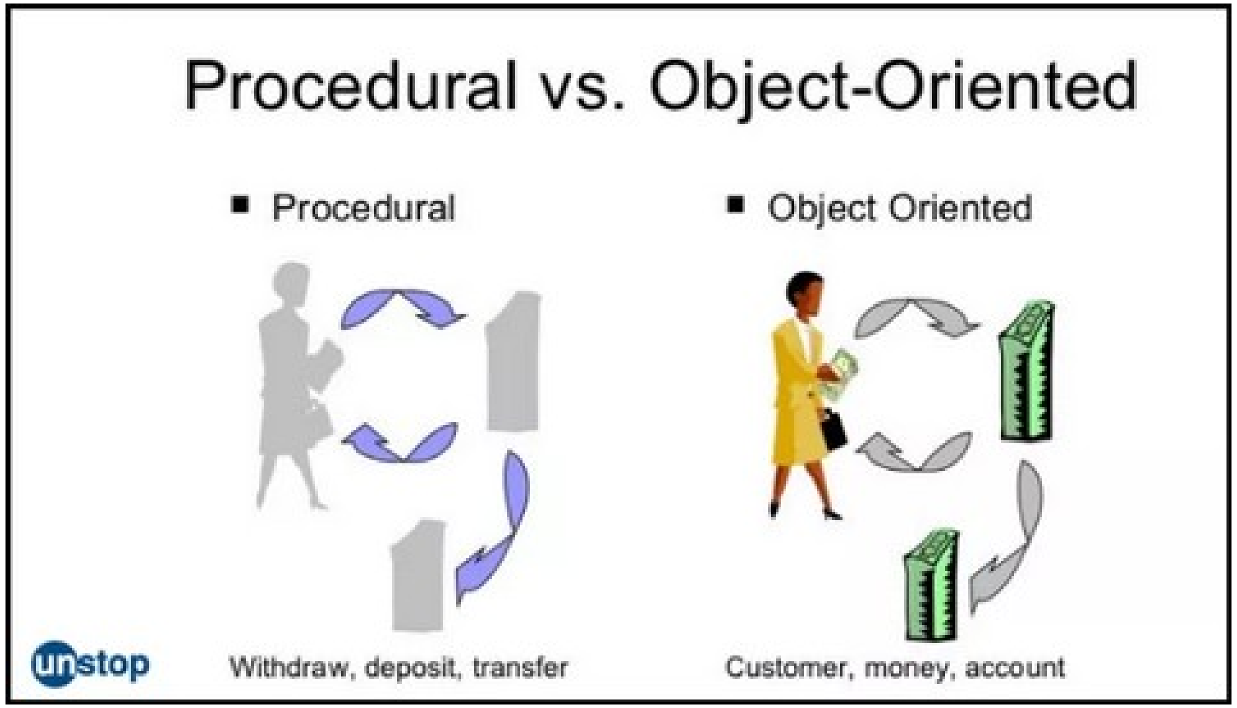
- C allows you to group several variables together into a single item known as a structure.

```
struct tag
{
    variable declaration;
    // ... more declarations
    //     may follow...
};
```

Practice: Example 4

Object oriented Programming (OOP)

- Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic.



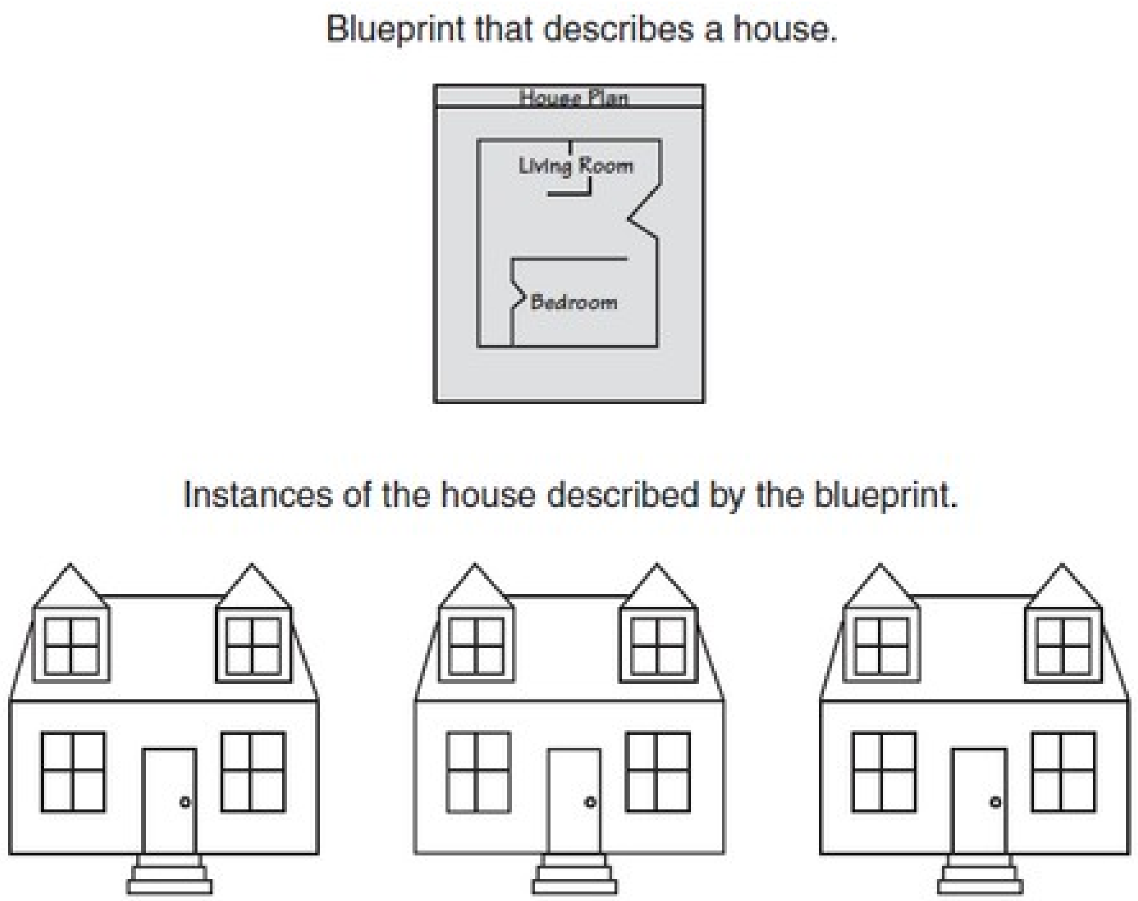
Object oriented Programming (OOP)

- An **object** is a software entity that contains both data and procedures.
- The data that are contained in an object are known as the object's attributes .
- The procedures that an object performs are called member functions.
- The object is, conceptually, a self-contained unit consisting of attributes (data) and procedures (functions).

Object oriented Programming (OOP)

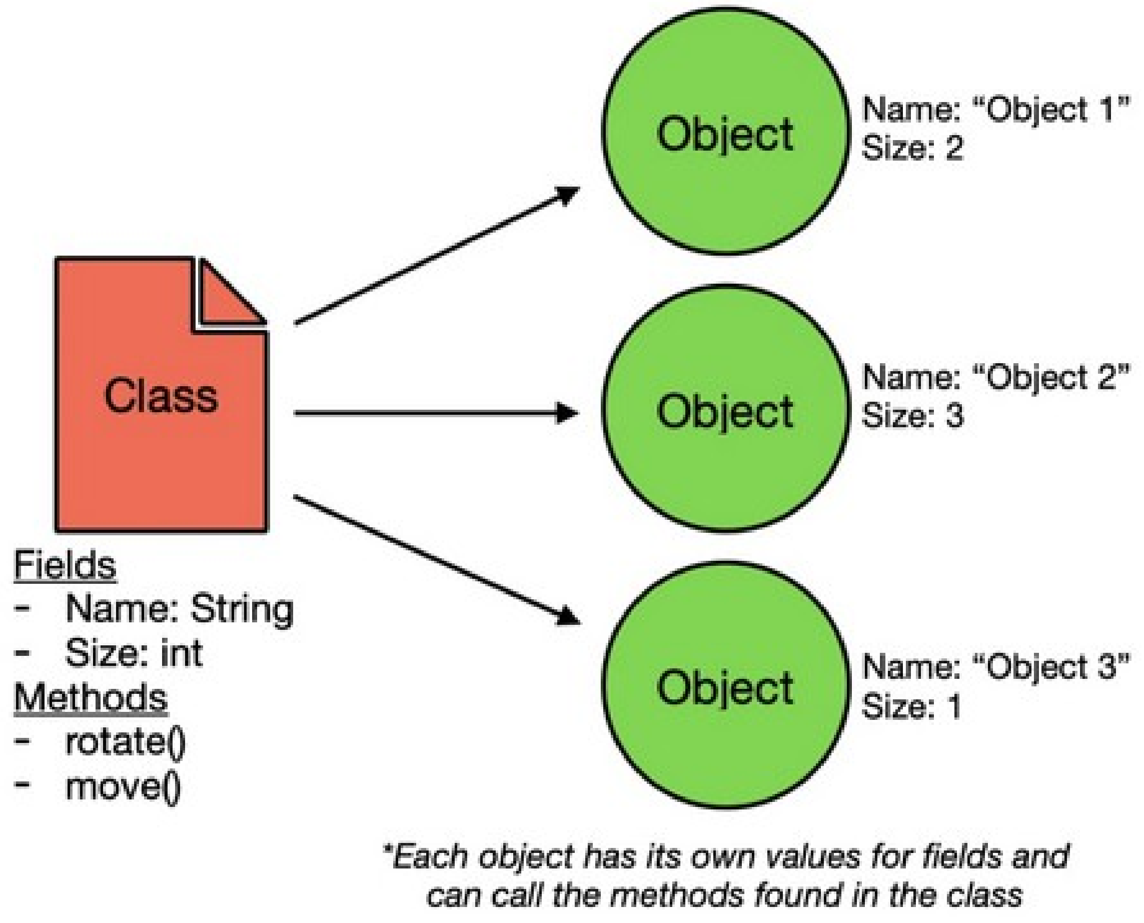
- A `class` is code that specifies the attributes and member functions that a particular type of object may have.

- Think of a `class` as a "blueprint" that objects may be created from. It serves a similar purpose as the blueprint for a house. The blueprint itself is not a house, but is a detailed description of a house.


Blueprint that describes a house.
House Plan
Living Room
Bedroom
Instances of the house described by the blueprint.

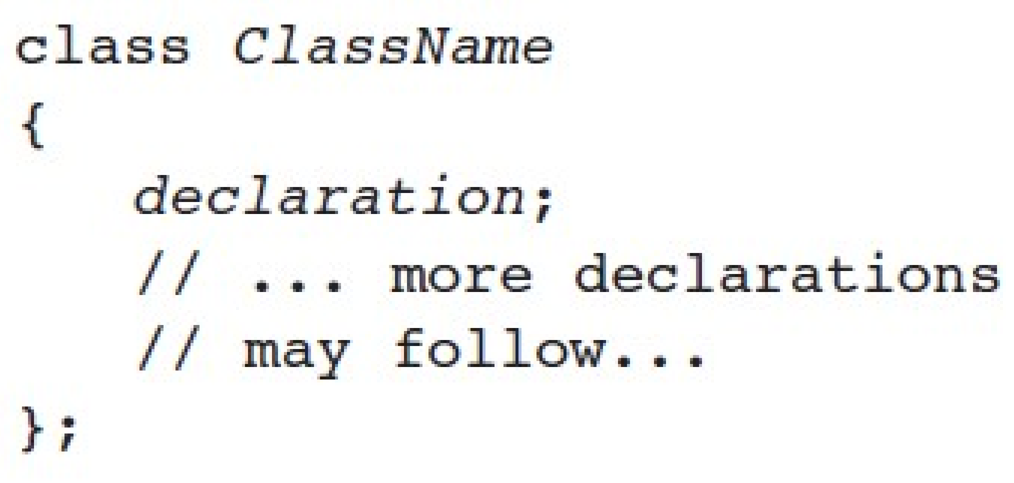
Object oriented Programming (OOP)

- A `class` is not an object, but it is a description of an object.
- When the program is running, it uses the `class` to create, in memory, as many objects of a specific type as needed.
- Each object that is created from a class is called an **instance** of the class.



Object oriented Programming (OOP)

- The general format of a class declaration is as below:



```
class ClassName
{
    declaration;
    // ... more declarations
    // may follow...
};
```

- The declaration statements inside a class declaration are for the variables and functions that are members of that class.

Practice: Example 5

Object oriented Programming (OOP)

- A **constructor** is a member function that is automatically called when a class object is created.
- A constructor is a member function that has the same name as the class.
- It is helpful to think of constructors as initialization routines. They are very useful for initializing member variables or performing other setup operations.

Practice: Example 6

Object oriented Programming (OOP)

In class Exercise

Create a class named Circle with a private member variable named radius. Write set and get functions to access the radius variable, and a function named calcArea that returns the area of the circle. You also need to write a constructor in your class.

Test your Circle class in the Main program by creating two instances of the class and use the calcArea to calculate the area of the two instances of Circle.

Practice: Exercise 1

See you next class