Lecture 7

Object-Oriented Programming III

Member Functions

Prof. Hyeong-Seok Ko
Seoul National University
Graphics & Media Lab

Contents

- Member functions (7.7, 12.2)

Member Functions

- Components of a member function
  - Return type
  - Function name
  - Parameters
  - Function body

```
#include <iostream>

class Box {
public:
   Box() {}
   void set(double h, double w, double l) {
      height = h; width = w; length = l;
   }
   double volume() const { return height*width*length; }

   double height, width, length;
};

void main() {
   Box box;
   box.set(1,1,1);
   std::cout << box.volume() << std::endl;
}
```

member functions

Member Functions

- const member functions
  - A const member function cannot change the content of the object for which the member function is called.

```
class Box {
public:
   Box() {}
   void set(double h, double w, double l) {
      height = h; width = w; length = l;
   }
   double volume() const { return height*width*length; }

   // volume() function may read but may not change the member variables

   double height, width, length;
};
```

Member Functions

- `const` member functions
  - A `const` member function cannot change the content of the object for which the member function is called.


```
class Box {
public:
   Box() {}
   void set(double h, double w, double l) const {
      height = h; width = w; length = l;
   }
   double volume() const { return height*width*length; }

   // volume() function may read but not write to the data members of the objects

   double height, width, length;
};

 void main() {
    Box box;
    box.set(1,1,1);
    std::cout << box.volume() << std::endl;
 }
```

Member functions

- `mutable` member variable
  - A `mutable` data member is a member that is never const.

```
class Box {
public:
   Box() {}
   void set(double h, double w, double l) const {
      height = h; width = w; length = l;
   }
   double volume() const { return height*width*length; }

   // volume() function may read but not write to the data members of the objects

   mutable double height, width, length;
};

 void main() {
    Box box;
    box.set(1,1,1);
    std::cout << box.volume() << std::endl;
 }
```

It works

Member Functions

- Default arguments in member functions

```
class Box {
public:
   Box() {}
   void set(double h, double w=1, double l=1) {
      height = h; width = w; length = l;
   }
   double volume() const { return height*width*length; }

   double height, width, length;
};

void main() {
   Box box;
   box.set(1);                                    // same as box.set(1,1,1)
   box.set(2,3);                                  // same as box.set(2,3,1)
   std::cout << box.volume() << std::endl;
}
```

Inline Member Functions

- A member function whose definition lies completely within the class definition. (e.g., `void set(…)`)
- The function definition can come elsewhere if an explicit inline declaration is made.

```
class Box {
public:
   Box() {}
   void set(double h, double w, double l) {
      height = h; width = w; length = l;
   }
   double area() const;
   inline double volume() const;

   double height, width, length;
};

double Box::area() const {
   return 2*(height*width + width*length + length*height);
}

inline double Box::volume() const {
   return height*width*length;
}
```

Defined within the class, thus inline

Declaration of an inline function

Definition of the inline function