CIS 014 – C++ Programming

Lecturer: Joseph Su

PASADENA
CITY COLLEGE®

REFERENCES

Optional Textbook:

Programming: Principles and Practice Using C++, 2nd ed, B. Stroustrup, Addison-Wesley, 2014

PDF:

http://www.cplusplus.com/files/tutorial.pdf

Online:

http://www.cplusplus.com/doc/tutorial/

The C++ Programming Language, 4th ed.

B. Stroustrup, Addison-Wesley, 2013

C++ How to Program, 10th ed

Deitel & Deitel, Pearson Hall, 2016

C++ Primer, 5th ed

S. Lippman, J. Lajoie, and B. Moo, Addison-Wesley, 2012

READING ASSIGNMENTS

ONLINE

- Classes
- C++ Classes and Objects
- Access Specifiers
- C++ Class Member Functions

TEXTBOOK

- Chapter 9 (up to 9.4.4 Defining member functions)

READING ASSIGNMENTS

REFERENCES

ASCII http://www.cplusplus.com/doc/ascii/
BOOLEAN http://www.cplusplus.com/doc/boolean/
RAND(): http://www.cplusplus.com/reference/cstdlib/rand/

http://www.cplusplus.com/files/tutorial.pdf (pages 1-94)
http://www.cplusplus.com/doc/tutorial/

- ✓ Program Structure
  - Complete all chapters
- ✓ Compound Data Types
- ✓ Classes
  - Classes (I)

TODAY

- Classes:
  - Idea & Concept
  - What They Are
  - Access Specifiers: Private, Protected, Public
  - Constructors
  - Member Access
  - Member Function Definitions (inside / outside of class)
  - Syntax/Anatomy

CLASSES: IDEA & CONCEPT

A Class:

- Represents a concept in a program
  - Think of “it” as a separate entity – an object in memory space
  - Examples: string, vector, matrix, input stream, robot, window screen, picture on screen, dialog box, etc.
- Is a user-defined type with its specific user-defined behaviors
- In C++, it is the building block for large programs

CLASSES: WHAT THEY ARE

- A class is a way of:
  - Encapsulating data
  - Defining abstract data types along with initialization conditions and operations allowed on that data
  - Hiding implementation details
  - Sharing behavior and characteristics

Class declaration:

```
class CRect {
  int x, y;
  public:
    void set_values (int,int);
    int area (void);
};
```

CLASSES: WHAT THEY ARE

- A class is a user-defined **data type**

- The following class’ name is `CRect`

```
class CRect {
   int x, y;

   public:
      void set_values (int,int);
      int area (void);
};

CRect rect;    // rect is a variable of type CRect
rect.x = 2;    // CANNOT access rect’s private member
               // variable x
rect.area();   // access rect’s member function area()
```

ACCESS SPECIFIERS: PUBLIC, PRIVATE, PROTECTED

- **public**: member variables and methods with this access specifier can be directly accessed from outside the class
- **private**: member variables and methods with this access specifier cannot be directly accessed from outside the class
- **protected**: member variables and methods with this access specifier cannot be directly accessed from outside the class with the exception of child classes

CLASS: CONSTRUCTORS

- Sometimes we need to ensure that certain variables in our object at run time has certain values before any member functions is called.

- We initialize these variables in the class’ constructor:

```
class CRectangle {
   int x, y;
   public:
      CRectangle(int, int);
      void set_values (int,int);
      int area () {return (x*y);}
};
```

// CRectangle(int, int) is only called when a CRectangle instance is created.

CLASS: CONSTRUCTORS

- If a custom constructor is available in a class, you use it.
- When `CRectangle rect(3,4)` is called in main(), an instance of the `CRectangle` class is created on the execution stack.
- That instance (or an object) is referred to by rect.
- When rect was created, `CRectangle’s` constructor was called:

```
CRectangle(int, int);
```

- If no constructor is explicitly called the program will invoke the default constructor:

```
CRectangle();
```

CLASS: MEMBER ACCESS

- Members of a class can be of various type.
- Data members (define representation of an object of the class)
- Function members (provide operations on such object)
- In C++:
  - Function members = member functions = methods (in other language context)
- Access to public members of an object – access notation:
  - [OBJECT].[MEMBER] if [OBJECT] is the actual instance allocated on stack
  - [OBJECT]->[MEMBER] if [OBJECT] is a pointer

- Examples:

```
CRect rect;      // rect is a variable of type CRect
rect.x = 2;      // CANNOT access rect’s member variable x
rect.area();     // access rect’s member function area()

CRect* rPtr = new CRect(); // rPtr is a pointer to
                           // the CRect instance
rPtr->x = 2; // CANNOT access instance’s member variable x
rPtr->area();// access instance’s member function area()
```

CLASS: MEMBER FUNCTION DEFINITION

- When we define a member function OUTSIDE of a class we need to tell which class it is a member of.
- For example, `set_values()` is a member of the `CRect` class we previous defined.
- When we define a member outside its class:

```
[CLASS_NAME]::[MEMBER_FUNCTION_NAME]
```

- Example:

```
void CRectangle::set_values (int a, int b) {
    x = a;
    y = b;
}
```

- Or you can use a member initializer list:

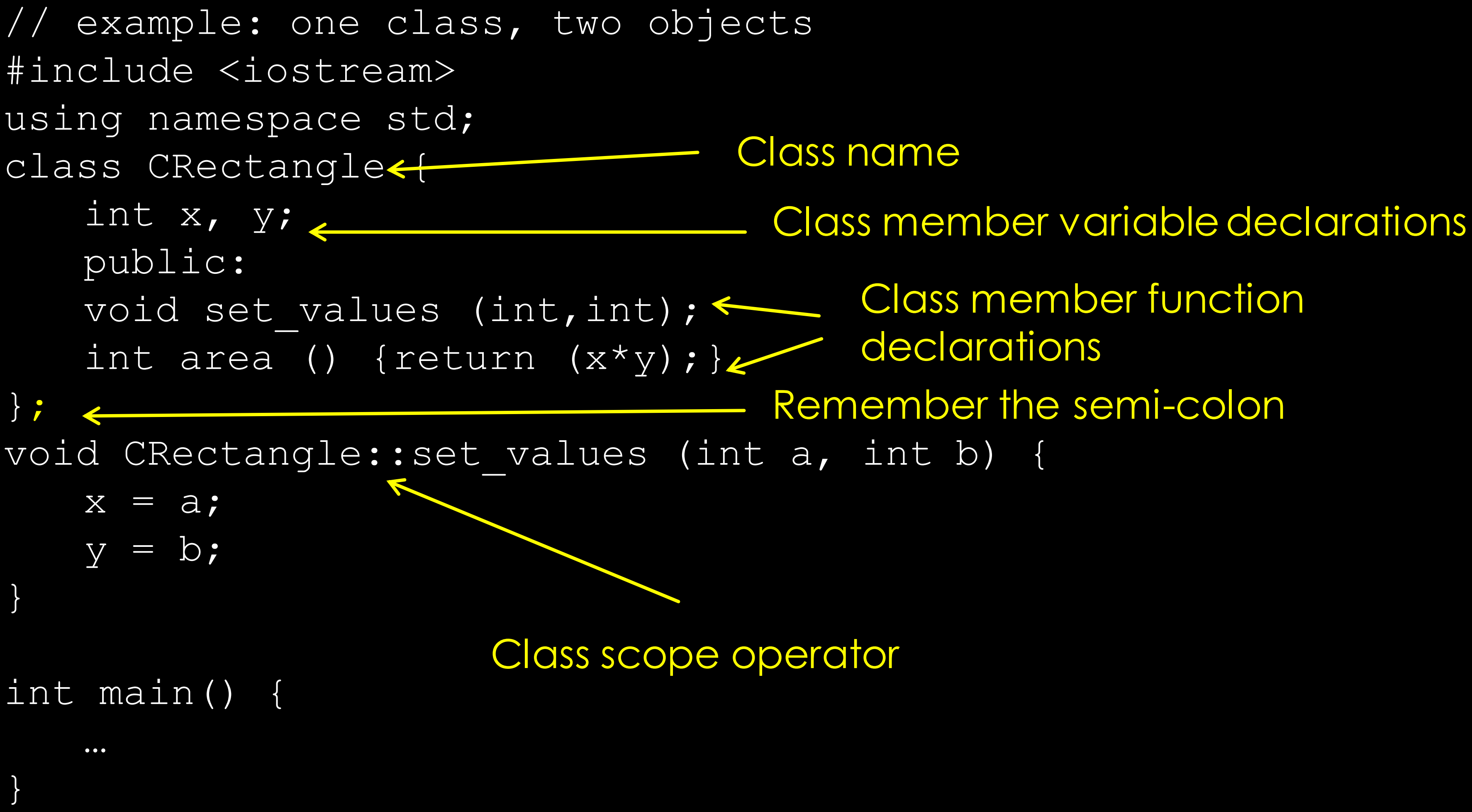
```
void CRectangle::set_values (int a, int b) :
    x{a}, y{b} {  // … }
```

CLASS: MEMBER FUNCTION DEFINITION

- When we define a member function INSIDE of a class we DON’T need to tell which class it is a member of
- For example, `area()` is a member of the `CRect` class we previous defined
- When defined inside a function definition a member function is *inline* (no function call instructions)
- *Inlining* a member function is recommended for a block of small expressions; larger code block should be defined outside of class
- Example:

```
class CRectangle {
   int x, y;
   public:
      void set_values (int a,int b) {x=a; y=b;}
};
```

CLASS: ANATOMY



```
// example: one class, two objects
#include <iostream>
using namespace std;
class CRectangle {
    int x, y;
    public:
    void set_values (int,int);
    int area () {return (x*y);}
};
void CRectangle::set_values (int a, int b) {
    x = a;
    y = b;
}

int main() {
    …
}
```

Class name

Class member variable declarations

Class member function declarations

Remember the semi-colon

Class scope operator

CLASS: ANATOMY

- Using the previously defined CRectangle class, we have

```
int main () {
    CRectangle rect, rectb;
    rect.set_values (3,4);
    rectb.set_values (5,6);
    cout << "rect area: " << rect.area() << endl;
    cout << "rectb area: " << rectb.area() << endl;
    return 0;
}
```

Two different instances of CRectangle

- Recall **main()** is your program’s entry
- We declare two instances of **CRectangle**, **rect** and **rectb**
- Each of **rect** and **rectb** is an object
- **rect** has its own life cycle, so is **rectb** having its own that is separate from **rect**’s