Foundation of Computer Science: Class

Kafi Rahman

Assistant Professor

Computer Science

Truman State University

13.2

Introduction to Classes

Class Example

```
class Rectangle
{
  private:
    double width;
    double length;
  public:
    bool setWidth(double);
    bool setLength(double);
    double getWidth() const;
    double getLength() const;
    double getArea() const;
};
```

Access Specifiers

- access specifiers are used to control access to members of the class. They can be
  - public: these members can be accessed in the program from outside of the class
  - private: these members can only be called by or accessed by functions that are members of the class

Class Example

```
class Rectangle
{
    private:

    public:

};
```

private Members

public Members

More on Access Specifiers

- Can be listed in any order in a class
- Can appear multiple times in a class
- If not specified, the default is private

Code Example

access specifier

Using const With Member Functions

- const appearing after the parentheses in a member function declaration specifies that the function will not change any data in the calling object.

```
double getWidth() const;
double getLength() const;
double getArea() const;
```

Defining a Member Function

- When defining a member function:
  - Put prototype in class declaration
  - Define function using class name and scope resolution operator (::)

```
int Rectangle::setWidth(double w)
{
        width = w;
}
```

Accessors and Mutators

- Mutator: a member function that stores a value in a private member variable, or changes its value in some way

- Accessor: function that retrieves a value from a private member variable.
  - Accessors do not change an object's data, so they are usually marked as const.

13.3

Defining an Instance of a Class

Defining an Instance of a Class

- An object is an instance of a class
- Defined like structure variables:
  - `Rectangle r;`
- Access members using dot operator:
  - `r.setWidth(5.2);`
  - `cout << r.getWidth();`

- We will get compiler error if we attempt to access private members by using dot operator
  - `r.width = 5.2; // will be an error`

Code Example

Rectangle class implementation

Program 13-1

```
1  // This program demonstrates a simple class.
2  #include <iostream>
3  using namespace std;
4
5  // Rectangle class declaration.
6  class Rectangle
7  {
8     private:
9        double width;
10       double length;
11    public:
12       void setWidth(double);
13       void setLength(double);
14       double getWidth() const;
15       double getLength() const;
16       double getArea() const;
17 };
18
19 //*****************************************************
20 // setWidth assigns a value to the width member.    *
21 //*****************************************************
22
23 void Rectangle::setWidth(double w)
24 {
25    width = w;
26 }
27
28 //*****************************************************
29 // setLength assigns a value to the length member. *
30 //*****************************************************
31
```

Rectangle Class (cont.)

```
32  void Rectangle::setLength(double len)
33  {
34     length = len;
35  }
36
37  //*****************************************************
38  // getWidth returns the value in the width member. *
39  //*****************************************************
40
41  double Rectangle::getWidth() const
42  {
43     return width;
44  }
45
46  //*******************************************************
47  // getLength returns the value in the length member. *
48  //*******************************************************
49
50  double Rectangle::getLength() const
51  {
52     return length;
53  }
54
```

Using: Rectangle Class

```
55  //*****************************************************
56  // getArea returns the product of width times length. *
57  //*****************************************************
58
59  double Rectangle::getArea() const
60  {
61     return width * length;
62  }
63
64  //*****************************************************
65  // Function main                                      *
66  //*****************************************************
67
68  int main()
69  {
70     Rectangle box;     // Define an instance of the Rectangle class
71     double rectWidth;  // Local variable for width
72     double rectLength; // Local variable for length
73
74     // Get the rectangle's width and length from the user.
75     cout << "This program will calculate the area of a\n";
76     cout << "rectangle. What is the width? ";
77     cin >> rectWidth;
78     cout << "What is the length? ";
79     cin >> rectLength;
80
81     // Store the width and length of the rectangle
82     // in the box object.
83     box.setWidth(rectWidth);
84     box.setLength(rectLength);
```

Using: Rectangle Class

```
85
86      // Display the rectangle's data.
87      cout << "Here is the rectangle's data:\n";
88      cout << "Width: " << box.getWidth() << endl;
89      cout << "Length: " << box.getLength() << endl;
90      cout << "Area: " << box.getArea() << endl;
91      return 0;
92  }
```

Program Output

```
This program will calculate the area of a
rectangle. What is the width? 10 [Enter]
What is the length? 5 [Enter]
Here is the rectangle's data:
Width: 10
Length: 5
Area: 50
```

Avoiding Stale Data

- Some data is the result of a calculation.
- In the Rectangle class the area of a rectangle is calculated as
  - area = length x width
- If we were to use an area variable (as a member) in the Rectangle class, its value would be dependent on the length and the width.
  - If we change length or width without updating area, then area would become stale.
- To avoid stale data, it is best to calculate the value of that data within a member function rather than to store it in a variable.

Pointer to an Object

- Can define a pointer to an object:
  - `Rectangle *rPtr;`
- Can access public members via pointer:
  - `rPtr = &otherRectangle;`
  - `rPtr->setLength(12.5);`
  - `cout << rPtr->getLength() << endl;`

Dynamically Allocating an Object

- We can also use a pointer to dynamically allocate an object.

```
// Define a Rectangle pointer.
Rectangle *rectPtr;

// Dynamically allocate a Rectangle object.
rectPtr = new Rectangle;

// Store values in the object's width and length.
rectPtr->setWidth(10.0);
rectPtr->setLength(15.0);

// Delete the object from memory.
delete rectPtr;
rectPtr = 0;
```

13.4

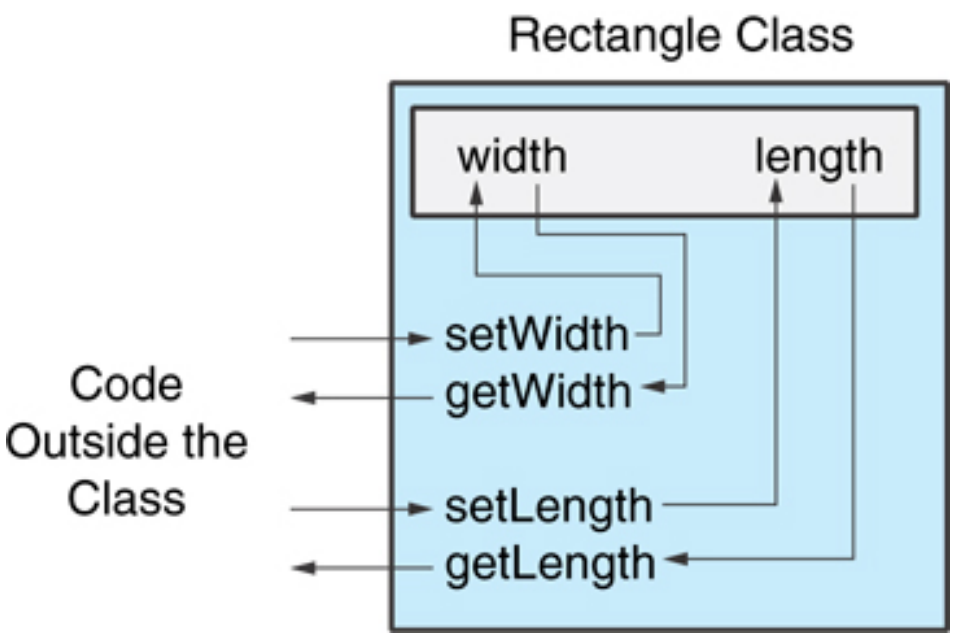
Why Have Private Members?

Why Have Private Members?

- Making data members private provides data protection
- Data can be accessed only through public functions
  - how does it protect data
  - let us modify the setWidth and the setLength function to see it in action
- Public functions define the class's public interface

Private Members of a Class

- Code outside the class must use the class's public member functions to interact with the object.



13.5

Separating Specification from Implementation

Separating Specification from Implementation

- Place class declaration in a header file that serves as the class specification file.
  - Name the file ClassName.h, for example, Rectangle.h
- Place member function definitions in ClassName.cpp, for example, Rectangle.cpp
  - File should #include the class specification file
- Programs that use the class must #include the class specification file. And that's it.