



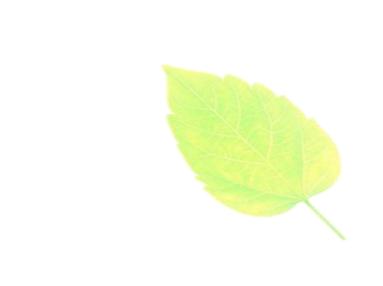
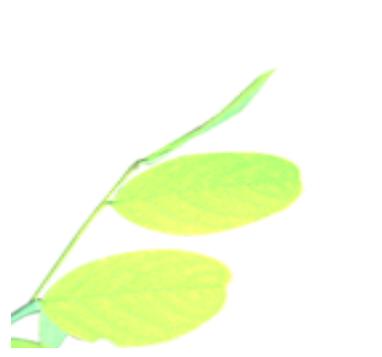
Introduction to C++ (Season 1)

Unit 4: Objects and Classes

第4单元: 物以类聚 - 对象和类

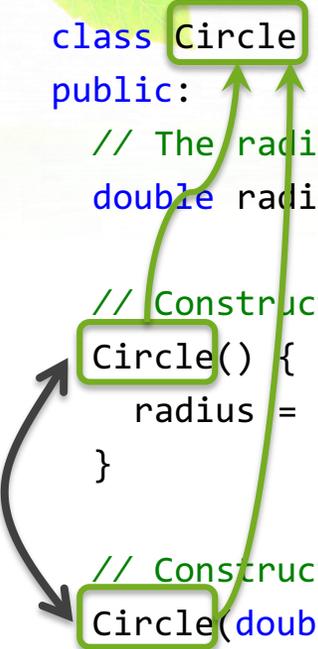
Section 2 : Create Objects and Access the members

第2节: 创建对象并访问对象成员



Constructors(构造函数)

```
class Circle {  
public:  
    // The radius of this circle  
    double radius;  
  
    // Construct a circle object  
    Circle() {  
        radius = 1;  
    }  
  
    // Construct a circle object  
    Circle(double newRadius) {  
        radius = newRadius;  
    }  
  
    // Return the area of this circle  
    double getArea() {  
        return radius * radius * 3.14159;  
    }  
};
```



Constructors:

- ❖ Initialize objects (构造函数: 初始化对象)
- ❖ Has the same name as the defining class (与类同名)
- ❖ NO return value (**including "void"**); (无返回值)
- ❖ constructors can be overloaded (可重载)
- ❖ may be no arguments (可不带参数)

A class may be declared **without** constructors (类可不声明构造函数)

1. A no-arg constructor with an empty body is implicitly declared in the class.
(编译器会提供一个带有空函数体的无参构造函数)
2. This constructor, called **a default constructor** is provided automatically **only if no constructors are explicitly declared in the class**.
(只有当未明确声明构造函数时, 编译器才会提供这个构造函数, 并称之为“默认构造函数”)

```
class Circle {  
public:  
    double radius;  
};
```



```
class Circle {  
public:  
    double radius;  
    Circle() { }  
};
```

Constructing Objects (创建对象)

❖ Without Arguments: (无参数)

```
ClassName objectName;
```

❖ For example:

```
Circle circle1; // the no-arg constructor  
                // is invoked
```

❖ With Arguments: (带参数)

```
ClassName objectName(arguments);
```

❖ For Example:

```
Circle circle2(5.5);
```

```
class Circle {  
public:  
    double radius;  
    Circle() {  
        radius = 1;  
    }  
    Circle(double newRadius) {  
        radius = newRadius;  
    }  
    //.....  
};
```

Object Member Access Operator(对象访问运算符)

Similar to struct variable

❖ To access the data & functions of an object: (访问对象中的数据和函数)

- the dot operator (.), namely, the object member access operator.

```
objectName.dataField // 访问对象的数据域  
objectName.function(arguments) // 调用对象的一个函数
```

instance variable (实例变量)

```
circle1.radius = 10;  
int area = circle1.getArea();
```

instance function (实例函数)

Question:

如果想用对象指针访问对象的数据和函数, 该用什么运算符?

A Simple Circle Class

```
#include <iostream>
using namespace std;
class Circle {
public:
    // The radius of this circle
    double radius;
    // Construct a circle object
    Circle() {
        radius = 1;
    }
    // Construct a circle object
    Circle(double newRadius) {
        radius = newRadius;
    }
    // Return the area of this circle
    double getArea() {
        return radius * radius * 3.14159;
    }
};
```

```
int main() {
    Circle circle1;
    Circle circle2(5.0);

    cout << "The area of the circle of radius " <<
        circle1.radius << " is " << circle1.getArea() << endl;
    cout << "The area of the circle of radius " <<
        circle2.radius << " is " << circle2.getArea() << endl;

    // Modify circle radius
    circle2.radius = 100.0;
    cout << "The area of the circle of radius " <<
        circle2.radius << " is " << circle2.getArea() << endl;

    return 0;
}
```

→ [TestCircle](#)