Module03 - C++ 编程语言

C++ 编程语言 - 课程目标

Module03 将完整的覆盖 C++ 编程语言的如下四个方面，作为一个 C++ Programmer ，熟知下列内容是必经之路：

- C++ 语言基础：类型和声明、指针与数组、表达式与语句、函数相关、作用域
- C++ 面向对象编程 (Object-Oriented Programming)：类、继承与多态、运算符重载、函数对象
- C++ 泛型编程 (Generic Programming)：模板 (Templates)
- C++ 标准库 (STL)：标准容器、迭代器、算法、 string 、 I/O 流

Module03-00
C++ 编程语言：概览

- 楔子
- 语言基础
- 面向对象编程
- 泛型编程

楔子 - 第一条线

在正式进入 C++ 语言学习之前，我们先随意看看几行 C++ 代码 (just for fun!)：

- 第一条线：看看过程式编程和泛型编程
  - Step1.1： 定义一个函数来对数组中每个元素加上 5（自定义函数）
  - Step1.2： 定义一个函数来对数组中每个元素减去 5，类似的需求如，乘以 5， ...
  - Step1.3： 试一下，用函数指针来简化工作？（函数指针）
  - Step1.4： 用标准库函数来完成工作（标准库函数）
  - Step1.5： 函数对象替代函数指针（函数对象）
  - Step1.6： 是该使用模板的时候了（模板）

楔子 - Step1.1 ： 函数

- 函数描述：
  - forEach() ： 为指定数组 a 中的元素加上 5
- 函数原型：

```
void forEach(int a[], const int& arraySize);
```

楔子 - Step1.2：更多需求、更多函数

- 更多需求：
  - 为指定数组的元素乘上 5；
  - 为指定数组的元素减去 5；
  - ……
- 需要更多的函数：

```
void forEach1(int a[], const int& arraySize); // a[i]+=5
void forEach2(int a[], const int& arraySize); // a[i]*=5
void forEach3(int a[], const int& arraySize); // a[i]-=5
// Many more...
```

楔子 - Step1.3：使用函数指针

- 操作的共性：
  - 加、减、乘等操作，都是用 n 对某些数组元素进行算术运算
- 简化 forEach 函数：

```
void add5(int& src);
void subtract5(int& src);
void multiply5(int& src);
//......
typedef void (*action) (int& src);

void forEach(int a[], const int& arraySize, action);
```

楔子 - Step1.4：使用标准库中的内容

- 现在我们使用标准库函数来解决问题：
  - for_each 函数

楔子 - Step1.5：函数对象

- 用函数对象替代函数指针：
  - action 函数指针的另外一种格式：函数对象
  - 如 add5() 对等的函数对象类似如下定义：

```
class Add {
public:
    Add();

    Add(const int& i);

    void operator()(int& src) {
        src += addValue;
    }
private:
    int addValue;
};
```

楔子 - Step1.6：使用模板

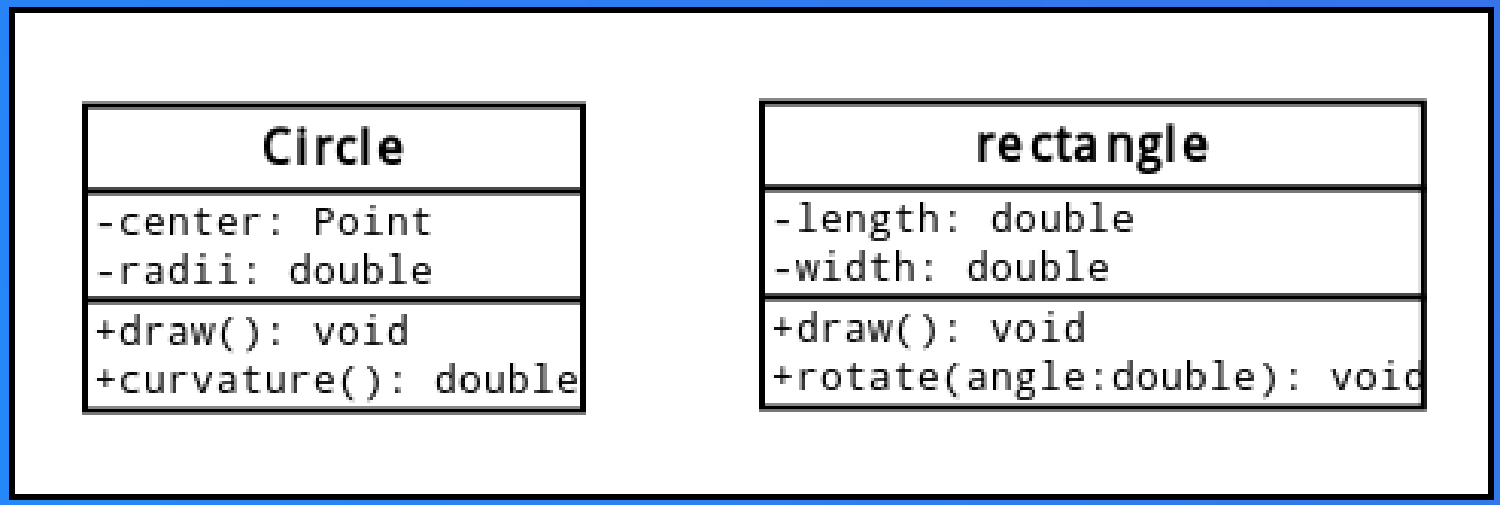
- add5() 等函数和 Add 等函数对象的显著局限：
  - 目前为止，只能处理 int 型的对象
  - 更多类型如 double， long 等是否意味着编写更多函数或函数对象？
- 模板 (template) 是解决上述问题的有效手段

楔子 - 第二条线

- 第二条线：类、继承、多态
  - Step2.1： 一些二维几何图形（类）
  - Step2.2： 很显然它们还有很多相同之处（基类、继承）
  - Step2.3： 尝试以下绘制过程（虚函数、多态）
  - Step2.4： 居然有些图形没有自己实现绘制方法！强制让所有图形都提供求 draw() 实现！（纯虚函数、多态）
  - Step2.5： 虽说几种图形混在一块，还是有办法区分的（运行期识别类型）

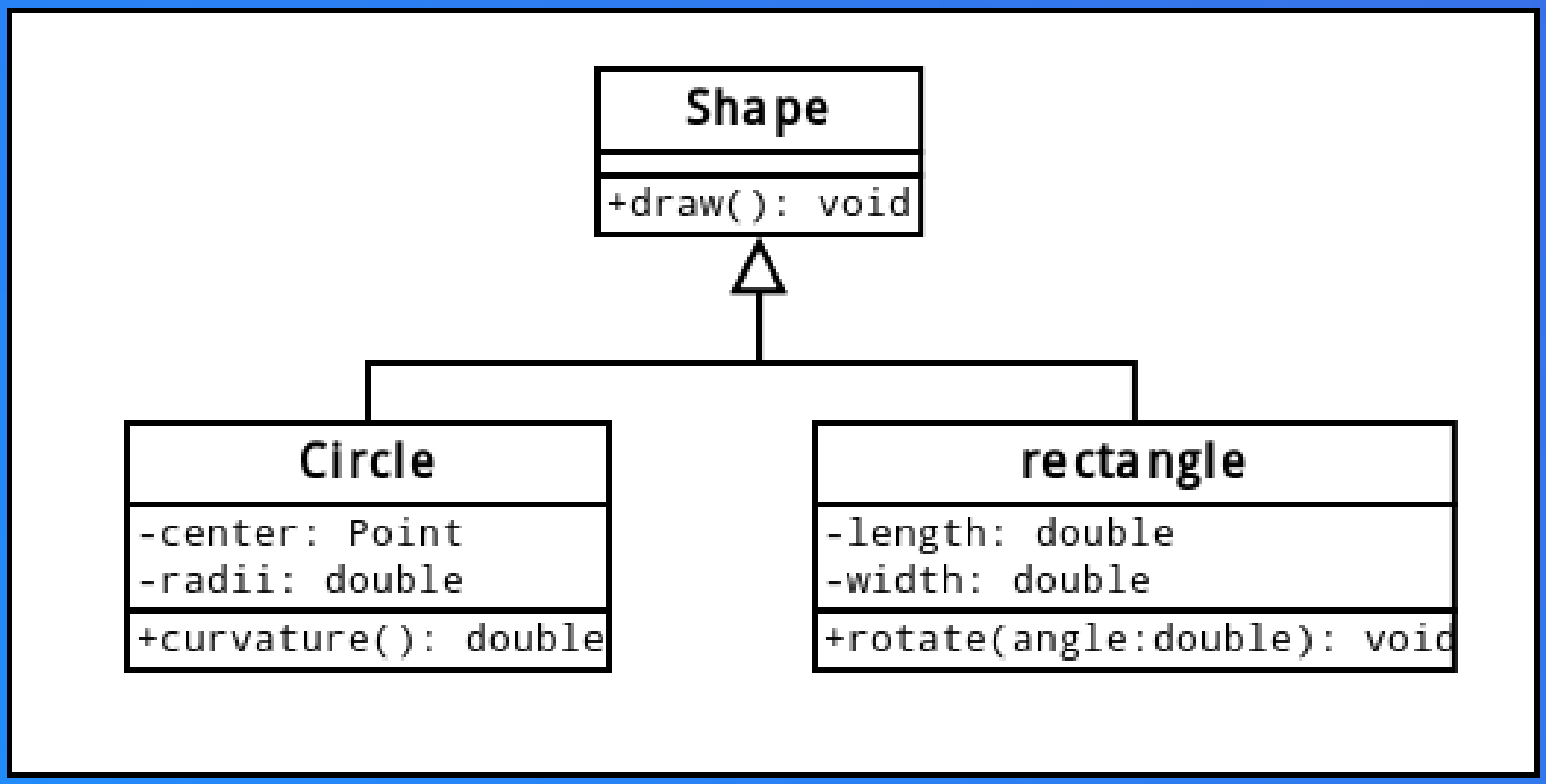
楔子 -Step2.1：一些图形类

- 两个简单的二维几何图形
  - Circle
  - Rectangle



楔子 -Step2.2： 提取共性

- 产生基类 Shape



楔子 -Step2.3： 虚函数的 overwrite

- 测试各种图形的 draw()

```
void testDraw(Shape* shape);
```

楔子 -Step2.4： 纯虚函数

- 将 Shape 的 draw() 指定为纯虚函数

```
void draw() = 0;
```

楔子 -Step2.5：运行期类型识别

- 通过 dynamic_cast 尝试着确定对象的类型