Swift

闭包:函数实际上是一种特殊的闭包:它是一段能之后被调取的代码。闭包中的代码能访问闭包所建作用域中能得到的变量和函数，即使闭包是在一个不同的作用域被执行的 - 你已经在嵌套函数例子中所看到。你可以使用{}来创建一个匿名闭包。使用in将参数和返回值类型声明与闭包函数体进行分离。

```
var numbers = [20,120,11,3]

    //匿名闭包

    let numbers2 = numbers.map({

        (number:Int) -> Int in

        let result = number * 3

        return result

    })

    print("改变后的数组\(numbers2)");
```

输出结果:**[60, 360, 33, 9]**

Swift 自动为内联函数提供了参数名称缩写功能，您可以直接通过$0,$1,$2来顺序调用闭包的参数。

如果您在闭包表达式中使用参数名称缩写，您可以在闭包参数列表中省略对其的定义，并且**对应参数名称缩写的类型会通过函数类型进行推断。** in关键字也同样可以被省略，因为此时闭包表达式完全由闭包函数体构成：

```
reversed = sorted(names, { $0 > $1 } )
```

在这个例子中，$0和$1表示闭包中第一个和第二个String类型的参数。

内联函数注意事项:

使用内联函数的时候要注意：

1.递归函数不能定义为内联函数

2.内联函数一般适合于不存在while和switch等复杂的结构且只有1~5条语句的小函数上，否则编译系统将该函数视为普通函数。

3.内联函数只能先定义后使用，否则编译系统也会把它认为是普通函数。

4.对内联函数不能进行异常的接口声明

内联函数和宏定义很像,但是宏有些限制:

但是宏也有很多的不尽人意的地方。

1、. 宏不能访问对象的私有成员。
2、. 宏的定义很容易产生二义性。

内联函数像宏一样的展开，所以取消了函数的参数压栈，减少了调用的开销。你可以象调用函数一样来调用内联函数，而不必担心会产生于处理宏的一些问题

****

swift的核心是面向协议编程,继承的子类思想其实对项目来说是不好的

- Swift的核心是面向协议的编程。
- 面向协议的编程的核心是抽象（abstraction）和简化（simplicity）。
- 所以swift的核心就是抽象和简化。

**********\

我认为不仅面向协议的编程（protocol oriented programming，后统一替换为POP）可以帮我们实现这点，另外2种编程类型也可以，且都具有抽象和简化的核心思想，这两种分别是：面向值的编程（value-oriented programming，后面统一替换成VOP）和函数式编程（functional programming）