钟业弘

1601214497

Ruby：

Ruby 是一种纯粹的面向对象编程的服务器端脚本语言，与 python 和 perl 类似。它的语法很简单，可拓展性强。Ruby 的语法和 C++很像，作为一门面向对象的编程语言，Ruby 有类和对象的概念。

Ruby 是动态类型语言。Ruby 的一切都是对象，从原始的字符串、数字，到类（class），都是对象。Ruby 的类是这样定义的：

```
class myclass
    ````
    ````
end
```

以 class 类名开头， 以 end 结尾，类的构造函数由 initial 方法定义。

在 Ruby 的类里，有四种变量：

局部变量：定义在类方法中的变量，作用域为定义该类的函数（或是函数内的语句块）。局部变量的定义和一般函数内的变量定义一致。

实例变量：类似于 c++中的类成员变量，在类的所有方法内可用，且不同类实例之间的实例变量不同。实例变量定义以@开头。

类变量：类似于 c++中的静态成员变量，在类的所有方法内可用，且不同类实例之间的类变量一致。类变量以@@开头。

全局变量：类似于 c++中的全局变量，在所有类之间可用。全局变量以$开头。

Ruby 的实例变量和类变量定义可以在类方法内完成，而 c++必须在类体里定义。

Ruby 的类与 c++很相似:

1. 都有访问控制（private,public,protect），但 Ruby 的 protect 类型只是类及其子类能访问，而 c++里还有友元类。
2. 都有类继承机制，但 Ruby 不支持多继承。
3. 都支持子类对父类函数的重载。

但 Ruby 有几个独特的特性：

1. to_s 方法：Ruby 的类可以定义 to_s 方法来返回对象的字符串表示，通过#{ myvar }得到 to_s 方法的返回值。
2. 冻结对象：Ruby 可以将对象冻结，使得对象的实例变量不能修改，具体使用方法如下：

```
x = myclass.new()          //这是类的实例定义
x.freeze                   //冻结对象
x.setvar 3                 //调用类方法去改变实例变量的值
```

在执行到第三条语句的时候，会报错，表示被冻结的对象 x 的实例变量不能被修改。

Scala:

Scala 是一门多范式（multi-paradigm）的编程语言，设计初衷是要集成面向对象编程和函数式编程的各种特性。

Scala 运行在 Java 虚拟机上，并兼容现有的 Java 程序。

Scala 源代码被编译成 Java 字节码，所以它可以运行于 JVM 之上，并可以调用现有的 Java 类库。

Scala 也有类和对象的概念，它的类定义如下：

```
class myclass(arg1: Int, arg2: Int) {
    var myvar1:Int = arg1
    var myvar2:Int = arg2
    `````
}
```

Scala 的类的构造函数直接在类定义的时候给出，这点是与 c++很不一样的。同时，可以通过定义 this 函数来定义其他的构造函数，但在定义这些构造函数时，必须基于已有的构造函数，如下面的定义

```
class myclass(arg1: Int, arg2: Int) {
    var myvar1:Int = arg1
    var myvar2:Int = arg2

    def this(arg:Int) {
        this(arg,0)
    }
    def myfunc() {
        println(“hello”)
    }
```
```

}

上面的代码块定义的类有 2 个构造函数，分别接受 1 和 2 个参数，在定义第二个构造函数时，必须调用第一个构造函数。

Scala 也有类的继承机制，但有 2 个需要注意的地方：

1. 重写父类的非抽象成员时要使用 override 关键字
2. 只有主构造函数（类体定义时的函数）可以往基类的构造函数里写参数，如下定义上面类的子类：

```
class mysubclass(arg1:Int, arg2:Int, arg3:Int) extends myclass(arg1,arg2) {
    var myvar3:Int = arg3
}
```

使用 extends 关键字声明父类，同时调用父类构造函数。

Scala 同样不支持多继承，且没有静态变量。

Scala 有一个 object 机制来实现静态变量的定义，它是一种单例化的对象定义，同名的 object 称为 class 的伴生对象，object 的定义必须是无参的，例子如下：

```
object myclass{
    var var1:Int = myclass.myvar1
}
```

此时 myclass 这个 object 可以访问类的成员变量 myvar1，且这个 var1 变量就是唯一的了，即可以作为 static 成员使用。