函数式编程结课报告

U201713321 CS1706袁杰威

1. 函数式编程的历史和发展

函数式编程的起源是Alan Turing的导师Alonzo Church给出的计算的数学模型lambda caculus。Lambda演算是一套从数学逻辑中发展，以变量绑定和替换的规则，来研究函数如何抽象化定义、函数如何被应用以及递归的形式系统。1936年Alan Turing提出了图灵机的设想，通过TAPE,HEAD,TABLE,STATE四个部分模拟人的纸笔运算，1945年冯诺伊曼提出了冯诺伊曼结构计算机，是通用图灵机的实现。图灵机设想和Lambda演算都是一种抽象计算模型，是解决可计算问题的一种方案，通俗来讲是将计算以数学的方式抽象化，以便证明和推导。图灵机和Lambda演算已经被证明是等价。

函数式编程是一种编程范式，常见的编程范式有命令式编程，函数式编程，逻辑式编程。

函数式编程是为了解决不确定性。一个函数，无论何时何地，给一个输入，它的输出永远确定，不受其他影响。最早的函数是编程语言是LISP，由John McCarthy在麻省理工学院，于20世纪50年代后期创造，在IBM 700/7000系列的科学计算机上。LISP语言主要有一下特性：

1. 用条件表达式来写递归函数
2. 函数做为参数传递

尽管LISP提出了很多 创新性的思想，但是在数学计算方面比FORTAN慢10到100倍。同时递归式的表达式，看起来没有命令式的直观易懂，所以LISP并没有应用于工业界。

继LISP后，出现了十多种函数式编程语言，其中最有名，范围最大的是：Haskell，第一个版本的Haskell于1990年发布。Haskell语言是在编程语言Miranda的基础上标准化的，并且以λ演算为基础发张而来。Haskell是现有的一门开放的、已发布的，且有多种实现的语言，支持惰性求值、模式匹配、列表解析、类型类和类型多态。

随着函数式编程的发展，主流函数式编程所具有的特性有以下几点：

1. 函数是“一等公民”：在支持函数式编程特性的语言里，函数被提到了一个非常重要的位置，他跟Int、String、Bool有着相同的地位。函数可以做为变量的字面量存储起来作为函数的参数和返回值在函数之间传递
2. 纯函数：相同的输入得到相同输出，函数的返回值和参数以外的其他隐藏信息或状态无关。
3. 只用“表达式，不用语句：表达式是一个单纯的运算过程，总是有返回值，“语句”是执行某种操作，没有返回值。函数式编程要求，只使用表达式，不实用语句。
4. 高阶函数：接受其他函数作为参数的函数有时被称为高阶函数，有了高阶函数，函数可以自由装配，由一些简单的函数装配出一些高级的函数。
5. 递归与尾递归：递归在几乎所有命令式编程语言中都有，即函数调用自身，在函数式编程中，由于没有可变状态，循环只能通过尾递归实现。
6. 惰性求值：表达式不在它被绑定到变量之后就立即求值，而是在该值被取用的时候求值
7. 柯里化：将一个多参数函数分解成多个单参数函数，然后单参数函数多层封装起来，每层函数都放回一个函数去接受下一个参数。

函数式编程的理论研究一直在进行着，但是几乎没有工业应用。其中主要的原因是函数式编程的效率不高，在那个计算资源稀缺的年代，在冯诺伊曼上实现的函数式语言没有早期的命令式语言快，这使得其发展速度慢于命令式编程语言，直到20世界80年代才有几种函数式编程语言应用在商业或工业系统。

在教育方面，函数式编程一直受到了很大的重视，很多学校使用函数式编程来教授算法和几何的相关概念。

1. 函数式编程的现状

由于函数式编程初期，CPU性能并不够好，函数式编程相对传统的OOP编程语言来说，效率相对较低。而随着摩尔定律逐渐失效，CPU的性能提升将体现在核数增加，这样并行的程序运行速度会越来越快，而并行程序最需要避免的事情就是赋值，而函数式编程的本质就是规避掉“赋值”，相比OOP编程的世界中，经常用到的技术是同步机制和加锁，而这通常意味着CPU算力的消耗。而由于函数式编程的特点，在函数编程的世界里就不会出现这个问题。这使得函数式编程又火了起来。

目前的主流OOP编程语言都已经添加了对函数式编程的支持，C++在C++11中添加函数式编程的相关接口。Java在Java8中添加了对函数式编程的支持。

C++作为一门多范式语言，支持面向过程，面向对象，函数式编程。C++的函数类型有三大类：函数指针，lambda和std::function。但是由于C++的历史包袱，虽然有这些可以实现函数式编程的效果，但是很难实现成熟的函数式编程。C++在进行静态类型函数式编程时，缺失几个必要特征：

1. GC，手动内存管理使得需要在代码中添加一些内存释放操作，不过针对这一点，C++提供智能指针，解决了部分问题。
2. 原生函数类型，C++为了支持函数类型，在标准库实现了类模版function，通过function模版类存储可调用的函数对象，本质上并不是原生函数类型。
3. 快捷方便的类型推导，C++通过decltype来实现，但是要搞出同等效果的类型推导，会影响编译的速度。
4. 延迟求值表达式流程控制，C++对表达式是立即求值的，没办法记录表达式的语法结构。

总的来说，C++作为历史悠久的面向对象编程语言，函数式编程的支持是建立在template系统、OO（面向对象）系统和一系列语法糖上面的，通过这些特性去模拟函数式编程特性，所以很难做到完美支持函数式编程。但是C++的哲学就并不是支持完整的函数式编程特性，为了支持完整的函数式编程而推倒一些其他语言特性得不偿失。

在Java的世界里，一起皆对象，如果想要调用一个函数必须要属于一个类或者对象，然后使用类或对象进行调用。面向对象编程编程可以降低代码耦合度，但是如果只是实现一些简单事情，面向对象编程会写很多重复的代码行。Java8中，引入了函数式编程与lambda表达式。Java对函数式编程的支持主要通过函数式接口（Functional interfacese）实现，在JDK中大部分函数式接口都会标记上@FunctionalInterface注解，这个注解主要是标记哪些是函数式编程接口，同时帮助编译器检查接口的合法性，Java的函数式接口中，只能有一个抽象方法。JDK提供了内置的API来表示函数。

JVM的GC使得函数式编程过程中不需要添加以下内存释放操作。但是和C++一样Java也是从原有的OOP基础上实现的函数式编程，很多函数式编程实际在编译的时候为了向下兼容，会被转化为OOP的代码。不过Java的强大还是在于OOP的完备性，不过函数式语言已经在JVM上崭露头角，如Scala和Clojure、Kotlin。

目前应用最广，最火的函数式编程语言便是JavaScript，虽然名字和Java很像，但是JavaScript和Java有着很大的区别，在JavaScript中，函数即对象，做到了函数是一等公民，对纯函数、柯里化、高阶函数都支持得很好。

除了Java和C++两门语言之外，从设计上讲，现代语言都或多或少吸收了函数式的一些特性，如function，闭包，尾递归，高阶函数等等。从设计理念上，函数式编程的无状态提供另外一种解决问题的方案。

1. 函数式编程课程总结

通过函数式编程课程的学习，首先对概念上有了一个清晰的认识，函数式编程关心的是类型之间的关系，而命令式编程关心解决问题的步骤。学习了函数式编程之后，我重新去看了一遍Java的函数式编程的设计，对其中的一些设计有了更加深刻认识，比如Java中的Lambda表达式引用外部变量要求为final，当时觉得很不方便，但是实际上这才是符合函数式编程的设计思想的，即：函数式编程不应为函数引入状态。

同时函数式编程的简洁性吸引到了我，所有在使用其他语言时，尝试使用函数式编程的方式去编写代码。

函数式编程思想拓展了我解决问题的思路，比如递归而不是迭代，函数注入而不是对象注入。不过我觉得，函数式编程编程虽然有很多优点，但是不应该为了“函数式”而编程。编程的目的是解决问题，命令式编程仍然会是主流趋势，但是在命令式编程中结合函数式编程是一个很好的方案，这样可以利用函数式编程的优势，使得代码更加简洁的同时，不损失其性能。

总的来说，函数式编程这门课托充了我的编程语言和思想的视野，之前学得都是类C语言，抛开不同语言的运行时机制，很大程度上只是语法上的区别，并没有给我提供一种解决问题的新思路。我对于主流语言的函数式编程的底层具体实现（如GC，内存管理等方面）还不够了解，后续会去深入了解下实现。