

**函数式编程原理课程报告**

|  |  |
| --- | --- |
| 姓 名： | 梁一飞 |
| 学 院： | 计算机学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1706 |
| 学 号： | U201714762 |
| 指导教师： | 郑然 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2020 年 4 月 20 日

目录

[1.函数式编程语言的家族成员简介 3](#_Toc37605557)

[1.1 ML 4](#_Toc37605558)

[1.2 LISP 4](#_Toc37605559)

[1.3 Haskell 4](#_Toc37605560)

[2.总结 3](#_Toc37605561)

[2.1 课程学习总结 3](#_Toc37605562)

[2.2 心得体会 4](#_Toc37605563)

# 1.函数式编程语言的家族成员简介

## 1.1 ML

ML是一个通用的函数式编程语言，它是由爱丁堡大学的Robin Milner及他人在二十世纪七十年代晚期开发的。它的语法是从ISWIM得到的灵感。作为元语言的ML是为了帮助在LCF定理证明机中寻找证明策略而构想出来的。（之前的元语言是pplambda，它联合了一阶逻辑演算和有类型的多态的λ演算）。它使用了Hindley-Milner类型推论算法来推测大多数值的类型，而不需要四处使用注解。

ML一般被归为非纯函数式编程语言，因为它允许副作用和指令式编程。这一点和纯函数式编程语言——例如Haskell——很不一样。

ML特性包括：传值呼叫（Call by value）的求值策略，一级函数，带有垃圾收集的自动内存管理，参数多态，静态数据类型，类型推论，代数数据类型，模式匹配和异常处理。

不像Haskell，ML使用及早求值，也就是说所有的子表达式总是被求值。导致的一个结果是你不能使用无穷表。然而，惰性求值产生的无穷表可以通过使用匿名函数来模拟。

今天在ML家族中有好几种语言：两种主要的方言是Standard ML和Caml，其他的包括F#－针对Microsoft .NET平台的开放研究项目。ML中的思想影响了众多的语言，例如Haskell，Cyclone和Nemerle。

ML的实力大多被用于语言设计和操作（编译器、分析器、定理证明机），但是它作为通用语言也被用于生化，金融系统，和宗谱数据库，一个P2P的客户／服务器程序等等。

## 1.2 LISP

Lisp（历史上拼写为LISP）是具有悠久历史的计算机编程语言家族，有独特和完全括号的前缀符号表示法。起源于公元1958年，是现今第二悠久而仍广泛使用的高端编程语言。只有FORTRAN编程语言比它更早一年。Lisp编程语族已经演变出许多种方言。现代最著名的通用编程语种是Clojure、Common Lisp和Scheme。

Lisp最初创建时受到阿隆佐·邱奇的lambda演算的影响，用来作为计算机程序实用的数学表达。因为是早期的高端编程语言之一，它很快成为人工智能研究中最受欢迎的编程语言。在计算机科学领域，Lisp开创了许多先驱概念，包括：树结构、自动存储器管理、动态类型、条件表达式、高端函数、递归、自主（self-hosting）编译器、读取﹣求值﹣输出循环（英语：Read-Eval-Print Loop，REPL）。

"LISP"名称源自“列表处理器”（英语：LISt Processor）的缩写。列表是Lisp的主要数据结构之一，Lisp编程代码也同样由列表组成。因此，Lisp程序可以把源代码当作数据结构进行操作，而使用其中的宏系统，开发人员可将自己定义的新语法或领域专用的语言，嵌入在Lisp编程中。

代码和数据的可互换性为Lisp提供了立即可识别的语法。所有的Lisp程序代码都写为S-表达式或以括号表示的列表。函数调用或语义形式也同样写成列表，首先是函数或操作符的名称，然后接着是一或多个参数：例如，取三个参数的函数f即为（f arg1 arg2 arg3）。

Lisp语言的主要现代版本包括Common Lisp, Scheme，Racket以及Clojure。1980年代盖伊·史提尔二世编写了Common Lisp试图进行标准化，这个标准被大多数解释器和编译器所接受。还有一种是编辑器Emacs所派生出来的Emacs Lisp（而Emacs正是用Lisp作为扩展语言进行功能扩展）非常流行，并创建了自己的标准。

## 1.3 Haskell

Haskell是一种标准化的，通用的纯函数编程语言，有非限定性语义和强静态类型。它的命名源自美国逻辑学家哈斯凯尔·加里，他在数理逻辑方面上的工作使得函数式编程语言有了广泛的基础。在Haskell中，“函数是第一类对象”。作为一门函数编程语言，主要控制结构是函数。Haskell语言是1990年在编程语言Miranda的基础上标准化的，并且以λ演算为基础发展而来。这也是为什么Haskell语言以希腊字母“λ”（Lambda）作为自己的标志。Haskell具有“证明即程序、命题为类型”的特征。

Haskell是现有的一门开放的、已发布标准的，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。它是一门纯函数编程语言，这意味着大体上，Haskell中的函数没有副作用。Haskell用特定的类型来表达副作用，该类型与函数类型相互独立。纯函数可以操作并返回可执行的副作用的类型，但不能够执行它们，只有用于表达副作用的类型才能执行这些副作用，Haskell以此表达其它语言中的非纯函数。

Haskell拥有一个基于Hindley-Milner类型推论的静态、强类型系统。Haskell在此领域的主要创新就是加入了类型类（type class），原本设想作为重载的主要方式，在之后发现了更多用途。

Haskell的主要实现GHC是个解释器，也是个原生代码编译器。它可以在大多数平台运行，GHC在并发和并行上具有高性能的实现能力，也有丰富的类型系统，如广义代数数据类型和类型族（Type Families）。

单子是一个抽象类型，可以表达不同种类的计算，包括异常处理、非确定性、语法分析以及软件事务内存，其中一个应用是用于表达副作用的类型。单子定义为普通的数据类型，同时Haskell也为其提供了几种语法糖。

Haskell有一个活跃的社区，在线上包仓库Hackage上有丰富的第三方开源库或工具。

# 2.总结

## 2.1 课程学习总结

学完了函数式编程这门课，我了解到函数式编程有如下特点：

（1）并行。

在函数式编程中，程序员无需对程序修改，程序就可以并发运行。程序运行期间，不会产生死锁现象。原因是通过函数式编程所得到的程序，在程序中不会出现某一数据被同时修改两次及以上的情况，同样的，两个不同的线程就更不用说了。由于函数式编程有这样的优点，导致了程序员完全不用花费精力去考虑增加某个线程带来的并发问题。

（2）不修改状态、无副作用。

在函数式程序语言中，所有的功能的经果就是一个返回值，不存在其他的行为，包括对外部变量的修改。在其他类型的语言中，变量是用来保存状态的。由于函数式编程不修改变量，导致了这些状态不能存在于变量中。那么，函数式编程语言保存状态的方法是使用参数来保存，递归方法是最好的例子。由于采用了递归方法，函数式编程语言在运行速度上相对

于其他语言较慢，所以，速度不够快是函数式编程语言长期不能广泛使用的主要原因。

（3）惰性求值

即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。惰性求值的一个好处是能够建立可计算的无限列表而没有妨碍计算的无限循环或大小问题。例如，可以建立生成无限斐波那契数列表的函数（经常叫做“流”）。第n个斐波那契数的计算仅是从这个无限列表上提取出这个元素，它只要求计算这个列表的前n个成员。

（4）柯里化

链接库往往将函数定义得比较一般化，具有通用性。这样的函数，需要传入比较多的参数。利用柯里化的方式，可以定义出“特殊化”的函数。所谓柯里化，是把接受多个参数的函数变换成接受一个单一参数（最初函数的第一个参数）的函数，并且返回接受余下的参数而且返回结果的新函数的方法。柯里化与部分求值是相关的，但不完全相同。

（5）模式匹配

模式匹配类比于命令式语言中函数的重载，即执行同一功能的函数可以根据输入参数类型和个数的不同做出不同的反应。从而实现分支功能。模式匹配可以让系统自动帮我们进行分支与变量的指定，有了模式匹配，函数式编程可以降低依赖switch/case而且写出来的程序代码也不会像switch/case那样一大块。

（6）递归

函数式编程中没有其他语言中while、for等循环，想要实现类似的功能只能通过递归实现。另外为了防止多重递归调用导致的爆栈等问题，函数式语言中往往会对递归进行优化，变化为等价的尾递归，某些编译器在编译器会对尾递归进行优化，使得每一次调用都使用相同的栈空间。

（7）高阶函数

在函数式编程中，函数也可以视为一个值，所以可以将一个函数作为另外一个函数的参数。一个函数就接收另一个函数作为参数，那么这么函数就是高阶函数。

使用函数式编程有如下好处：

（1）更方便测试

编写可测试的代码是当下软件开发行业的基本要求，代码在交付前要编写测试用例对每一个功能做单元测试。副作用增加了代码测试难度，原因在于当你尝试测试依赖于外部条件的函数，测试用例也必须运行在相同的条件下。纯函数不依赖也不改变外界的状态，只要给定输入参数，返回的结果必定相同。因为没有副作用，所以不必提供特定的单元测试环境、编写单元测试将变得更加简单、轻松、

（2）易于并行编程

编写多线程代码尤其注意资源的互斥访问及死锁的问题。当纯函数在多线程环境执行的时候，不必担心互斥与死锁。纯函数不会去读取或修改外部对象的值，因此不必编写互斥或者读写锁之类的代码来保证资源被有序访问。

（3）开发快速

由于高阶函数的特性，多个函数可以组合成一个更强大的函数。柯里化技术主张用现有函数来构建新函数的理念。这极大地提高了代码的复用程度。以Lisp和C的比较为例，实现相同的功能，C代码的长度是Lisp的7倍到10倍。

## 2.2 心得体会

因为之前没有学习过函数式编程，所以刚刚接触上手困难，以前都是面向过程，面向对象编程，往往有太多的语法规则束缚，而函数式编程简练的写法令我大感新奇。老师教学认真，在群里及时解答问题，课程体验良好。对语法熟悉后就简单了，这门课的实验内容采用ML语言完成一些简单的操作，如果想到思路，很快就能解决。我对于函数式编程的理解是这样的：对于一个问题，给定一个输入，我们要编写一个函数作用于这个输入，然后得到一个中间结果，将这个中间结果输入到下一个函数，依此下去，直到得到我们所需的最终结果，程序员的任务就是实验初始输入到输出中间的所有函数。