大作业选题报告

付祈安　基科 52　2015012200

2016 年 11 月 18 日

目录

1 选题内容

实现一个简单的类 Lisp 编程语言的解释器，在此基础上，提供一个交互式的命令行。

2 选题背景意义

2.1 理论意义

通过自己写一个编程语言的解释器，可以了解一门语言的解释器是如何工作的，比如变量的作用域是如何实现的，内存是如何管理的，函数调用是如何返回的，递归地调用一个函数是如何实现的等等。

同时，由于这将是一个函数式编程语言，通过写这样一个解释器，也有助于自己了解函数式编程语言。

2.2 实用性

一个解释器可以提供一个交互式的命令行，日常生活中，可以利用脚本语言易于调试、可以通过交互式的命令行来验证自己的想法的特性来编写一些小的脚本方便日常生活。

3 同类软件调研分析

本软件的定位是：一门编程语言的解释器，并且语法的形式上模仿 Lisp，所以同类软件大致有如下两类：

3.1 解释型语言

解释型语言中，比较著名的有 Python、Ruby 等，但是常见的解释型语言少有类似 Lisp 语法的，或者说，少有函数式的解释型的编程语言，这是本软件的新意之一。

解释型语言的方便之处在于可以交互式的运行一段代码，本软件也将提供一个简单的交互式的命令行，所谓简单，是指将没有自动补全这种高级特性。

3.2 Lisp 系语言

Lisp，最初被拼为 LISP，一个历史悠久的电脑编程语言家族。最早由约翰 · 麦卡锡在 1958 年基于 λ 演算创造，演化至今，是历史第二悠久的高级语言，仅次于 Fortran，也是第一个函数式编程语言。

其名称源自列表处理器 (英语：List Processor) 的缩写。LISP 有很多种方言，各个实现中的语言不完全一样。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。

但是这些 LISP 语言大多是编译型的，因此写一个解释型的类 Lisp 语言是有意义的。但是由于时间限制，本软件的自定义语言将不会遵守 Common Lisp 标准。

此外，计划在这个解释器中实现一个无穷精度的整数类型，这也是大多数 Lisp 系语言所不具有的特性。

4 功能设计

4.1 语法形式

语法形式采用类似 Lisp 语言的形式，即：

```
(+ 1 2) #表示1+2
(assign a 3) #表示 a=3(即赋值语句)
```

最后一个语句定义了一个叫做 func 的函数，参数为 paraa 和 parab，返回值是 paraa + parab + 1。选择这样的语法形式的原因在于这样的语法易于写 parser。更具体的语法没有完全构思好 (一些细节的问题)。在完成大作业后会有详细的语法的说明和程序的实例。

这一语言的解释器将会支持自定义函数，比如：

```
(define func (paraa parab)
           (assign paraa (+ paraa 1))
           (+ paraa parab)
)
```

这段程序定义了一个名为 func 的函数，它有两个参数，分别是 paraa 和 parab，第二到三行是函数体，函数的返回值是函数体最后一行的表达式的返回值 (这也是很多函数式编程语言采取的做法)，也就是 (+ paraa parab) 的返回值，调用这个函数的格式为：

```
(func 1 2)
```

这将会返回 4，函数调用时，参数不仅可以是具体的数字，也可以是变量，例如：

```
(assign a 1)
(assign b 2)
(func a b)
```

此时函数的返回值依然是 4。

参数还可以是另一个表达式，例如：

```
(func (+ 1 2) (+ 1 3))
```

此时函数的返回值是 $(1+2)+1+(1+3)=8$。

4.2 基本功能概述

1. 会支持基本的数据类型，比如整数类型，布尔类型，在时间充裕的情况下也会实现字符串类型和链表类型 (类似于 Python 中的链表)。其中整数类型是无穷精度的。

2. 会支持简单的自动内存管理：每个变量名在被赋值的时候，会创建一个数据，并且该变量名被指向这个数据，每个数据内部会记录自己被这样引用了多少次，当引用次数为 0 时，这个数据所占用的内存空间就会被清空。并且提供一个内置的 None 数据，在需要清理内存的时候，将某个变量指向这个 None 数据，如果它原来指向的数据没有被其他变量引用，这个数据占用的内存就会被清空，从而达到清理内存的效果。

3. 会支持自定义函数和递归地调用函数，如果时间允许还会支持 lambda 表达式特性。调用函数时，传递参数传的是引用，返回时，传递的是函数体最后一个表达式的返回值的拷贝。

4. 会支持不同的变量作用域。每个括号表达式拥有一个变量的环境，当在一个表达式里使用一个变量时，程序会首先查找在当前的括号表达式的变量环境中有没有这个变量，如果没有，则到上一层的括号表达式的变量环境中查找这个变量，如此递归地执行，直到找到这个变量为止 (或者在最顶层的变量环境中也没有找到这个变量，此时则抛出异常)。对于函数，每个函数拥有单独的变量作用域，也就是说，函数不能访问调用它的表达式中的变量，反过来，函数运行结束后，在函数体中定义的变量以及参数中的变量都会被清空，因此调用一个函数后，在主调的表达式中无法访问函数中的变量。

5. 会实现一个简单的交互式命令行。也就是可以动态地输入一些代码并查看代码运行的结果，但是不会实现一些更高级的特性，比如自动补全。

6. 常见的 Lisp 系语言大多可以在代码中动态地定义一个函数并返回这个函数，作为一个高级特性，在时间充裕的情况下，本软件也将会支持。