**编译原理 实验报告**

|  |  |  |
| --- | --- | --- |
| **评 语（4号楷体）** | **成绩** |  |
| 教 师： xx  年 月 日 | | |

**学院班级： xxxxx xxx**

**学生学号： xxxxxxxxxxxx**

**学生姓名： xxxxxx**

**实验日期： 2018.12.22**

1. **实验目的**
2. 通过做上机题加深对编译器构造原理和方法的理解，巩固所学知识。
3. 会用正规式设计简单语言的词法；
4. 会用产生式设计简单语言的语法；
5. 会用递归下降子程序编写语言的解释器。
6. **实验环境**

Eclipse JDk8

1. **实验内容**
2. **题目要求**
3. 题目：为函数绘图语言编写一个解释器
4. 输入：用函数绘图语言编写的源程序
5. 用词法分析器识别其中的记号（可将记号的信息显示出来）；用语法分析器识别记号流中的语句（可将语句结构显示出来）；解释器：词法分析、语法分析、语义分析/计算，绘制图形
6. **设计以及实现的思路：**
7. 根据题目要求，经过自己的分析，我在实现解释器的程序中，将总体过程分为4个阶段，分别为**Scanner(词法分析)、Paerser(语法分析)、Semantic(语义分析)、Main(绘制图形与窗口建立),全过程采用JAVA语言、递归下降子程序方法以及增量模型开发思想完成解释器**。同时还为编译器**增加了文本框(使得用户可以在图形中添加文字说明)、规定图形颜色(红色或者黑色)、规定点的大小以及可调节的绘制图形速度。**
8. 在我设计的解释器中，我以**语义分析接口**作为整个程序的入口(实现语义分析的代码块继承了语法分析的代码块，同时采用“语法制导翻译”方法，这样就能做到两个过程合为一个阶段，即在语法分析的过程中，执行语义分析)，词法分析器作为语法分析器的子程序--为语法分析提供操作接口(获取记号)。
9. **词法分析**：
10. 词法分析器的主要功能是识别记号，在实现词法分析的过程中，我将词法分析的实现分为3个过程，分别为InitScanner(词法分析程序入口)、GetToken(词法分析主体程序)、CloseScanner
11. 词法分析的记号种类一共有6种，所以我利用Token\_Type.java记录这些记号种类。记号的组成包括记号的类别与属性，我利用面对对象思想，将记号看成一个对象，并用Token.java记录。由于我们完成的编译器的记号数目有限，所以我利用Token\_Table.java这个符号表存储所有可能出现的记号。(包括我自己实现的用户可规定图形颜色的3个记号，COLOR、RED、BLACK)
12. 在记号中由于包括6个数学函数公式，为了以后方便语义分析进行计算，我将这6个记号的其中一个属性设置为Func接口，每当识别到该种类型的记号，就会自动调用相应数学函数的接口。
13. 在实现词法分析的主体程序中，首先必须进行预处理，即过滤程序中的空格、回车、TAB符等。其次开始读取字符，若字符是A-Za-z，则它一定是函数、关键字、PI、E等;若字符是一个数字，则一定是一个常量;若字符不是字母和数字，则一定是符号；在判断的过程中都采用了最大化匹配字符的原则，同时还会判断识别出来的字符串是否在字符表中，若不在则返回ERRTOKEN。
14. **语法分析**：
15. 语法分析器的主要任务是分析语言的结构，同时利用递归下降分析为结构正确的输入构造语法树。
16. 根据函数绘图语言的文法，一开始可分为Program、Statement、OriginStatment、ScaleStatment、RotStatment、ForStatment、ColorStatement、Expression，但是此文法存在二义、左递归、左因子等缺点。继续改写此文法，在原来的文法基础上添加Term、

Factor、Component、Atom等即可消除这些缺点。然后将产生式改写为EBNF形式，根据老师的PPT，即可写出相应的递归下降分析程序。

1. 根据表达式的特点，我将表达式语法树的节点分为4类。

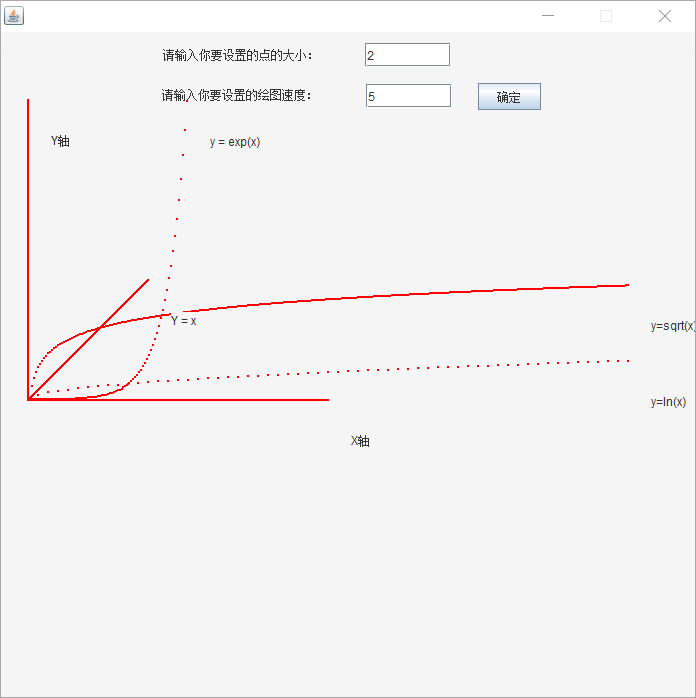
叶子结点(变量T)、叶子结点(常数)、两个孩子的内部结点(二元运算)、一个孩子的内部结点(函数以及函数地址)。利用面向对象的思想将此结构存入名为TreeNode的类中。

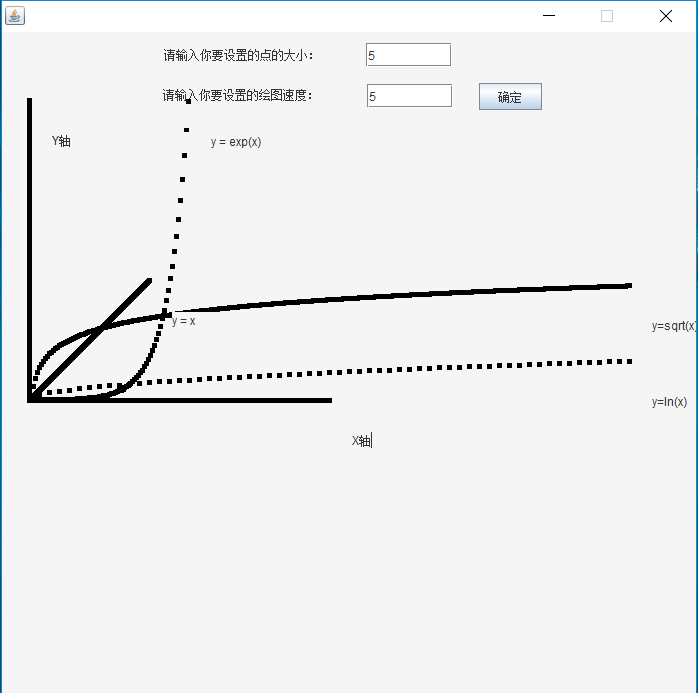
1. 在匹配OriginStatment、ScaleStatment、RotStatment、ForStatment、ColorStatement这些语句的时候，根据其句子结构的特点判定输入的结构是否正确。而在匹配Term、Factor、Component、Atom、Expression这些语句的时候，就必须根据自身的结构特点调用TreeNode类中相应的构造函数方法进行语法树的创建。
2. **语义分析**：
3. 语义分析器的主要任务是：根据语言结构，处理函数绘图语言程序的语义。根据语法分析构建的语法树，采用深度优先后序遍历即可计算出表达式的值，其次还可以在图形界面中画出每个坐标点。
4. 我实现的语义分析器是继承语法分析器，因为语义分析中需要计算表达式值的语句只有OriginStatement、ScaleStatement、ForStatement、RotStatement这四种，所以继承语法分析器的时候我只需要重写这4中类型的语句在利用GetTreeValue(TreeNode root)即可获得表达式的值。
5. 为了能够在用户交互界面中画出相应的点，我添加了额外的辅助函数：**CalCoord(TreeNode hor\_ptr,TreeNode ver\_ptr,Main.Point point)** //计算点的坐标值：首先获取坐标值，然后进行坐标变换

**DrawLoop(double start\_val,double end\_val,double step\_val,TreeNode x\_ptr,TreeNode y\_ptr)**循环绘制所有点。

**DrawLoop(start\_val, end\_val, step\_val, x\_ptr, y\_ptr);**绘制一个点

1. **绘制图形：**



****

1. 用JAVA插件windowbuilder绘制与用户交互的界面。
2. 运行程序后用户首先输入需要设置的点大小和绘图速度信息，点击确定，程序开始做图，并在结束生成文本编辑框。
3. 设置点的大小通过Graphics2D对象的内置函数setStroke();设置绘图速度通过线程休眠进行完成Thread.sleep((long)time)；设置文本框则是根据每个循环做图的最后一个点的坐标来确定文本框位置。
4. **心得体会**
5. **优点**：我编写的编译器除了基本的编译功能之外，同时能提供给用户自己设置绘制图形的颜色、绘制图形的点的大小、绘制图形的速度以及在绘制结束后还能生成相应的文本框，方便用户添加描述语言。
6. **缺点**：生成文本框的位置不佳，同时想要再次改变点的大小和绘图速度就必须重新启动程序。这些缺点都有待改善。
7. 在根据EBNF的文法生成递归下降分析程序的过程中，对于相应符号的转换掌握不够熟练，最后通过翻书和学习老师PPT解决此问题。
8. 构造语法树与遍历语法树的过程中，对于递归的理解不够深，直接导致我的程序在运行这一块的时候出现崩溃的情况，后来经过网上查阅资料与自己的Debug才将此问题解决。
9. 在绘制相应坐标点的时候一开始毫无头绪，后来发现点其实就是首尾坐标相同的线段，然后查阅JAVA API得到Graphics2D的使用方法。
10. 经过此次实验，觉得自己对于整个编译器的流程更加熟悉了，同时也在实现的过程中提高了自己编写代码的能力。