**编译原理 实验报告**

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

**教学班级： 1张立勇班**

**学生学号： 20009200591**

**学生姓名： 李傲松**

**实验日期： 2022.12**

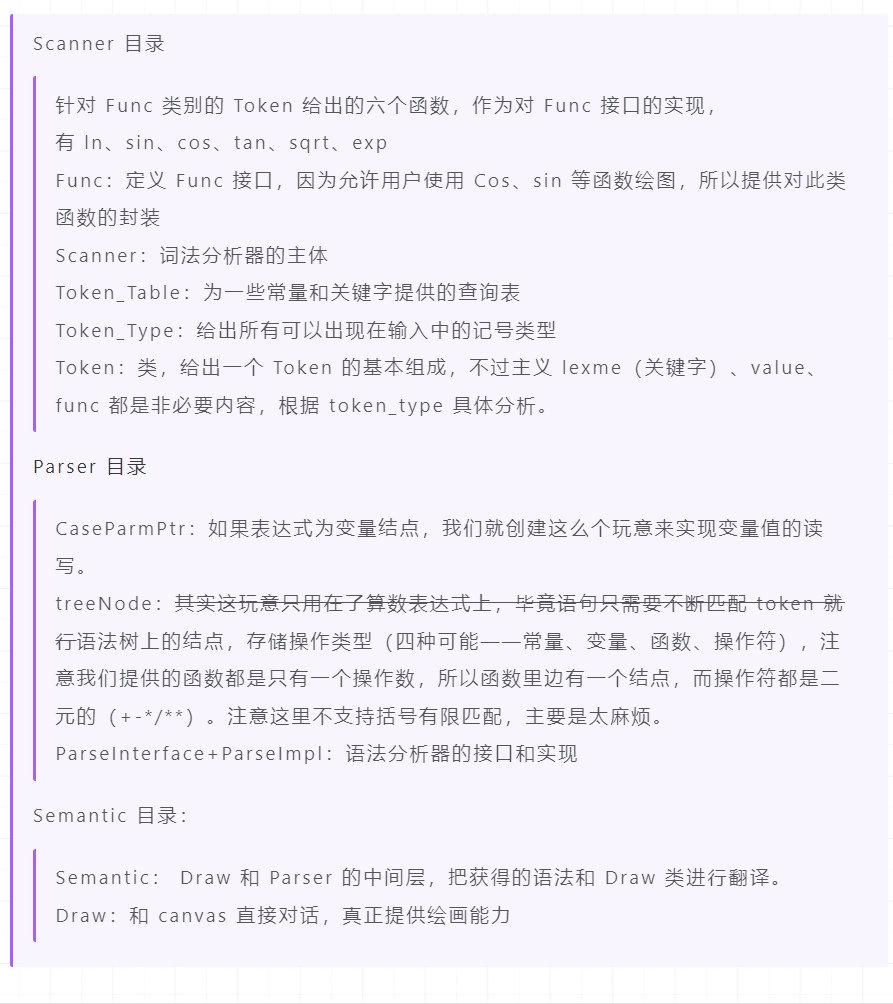
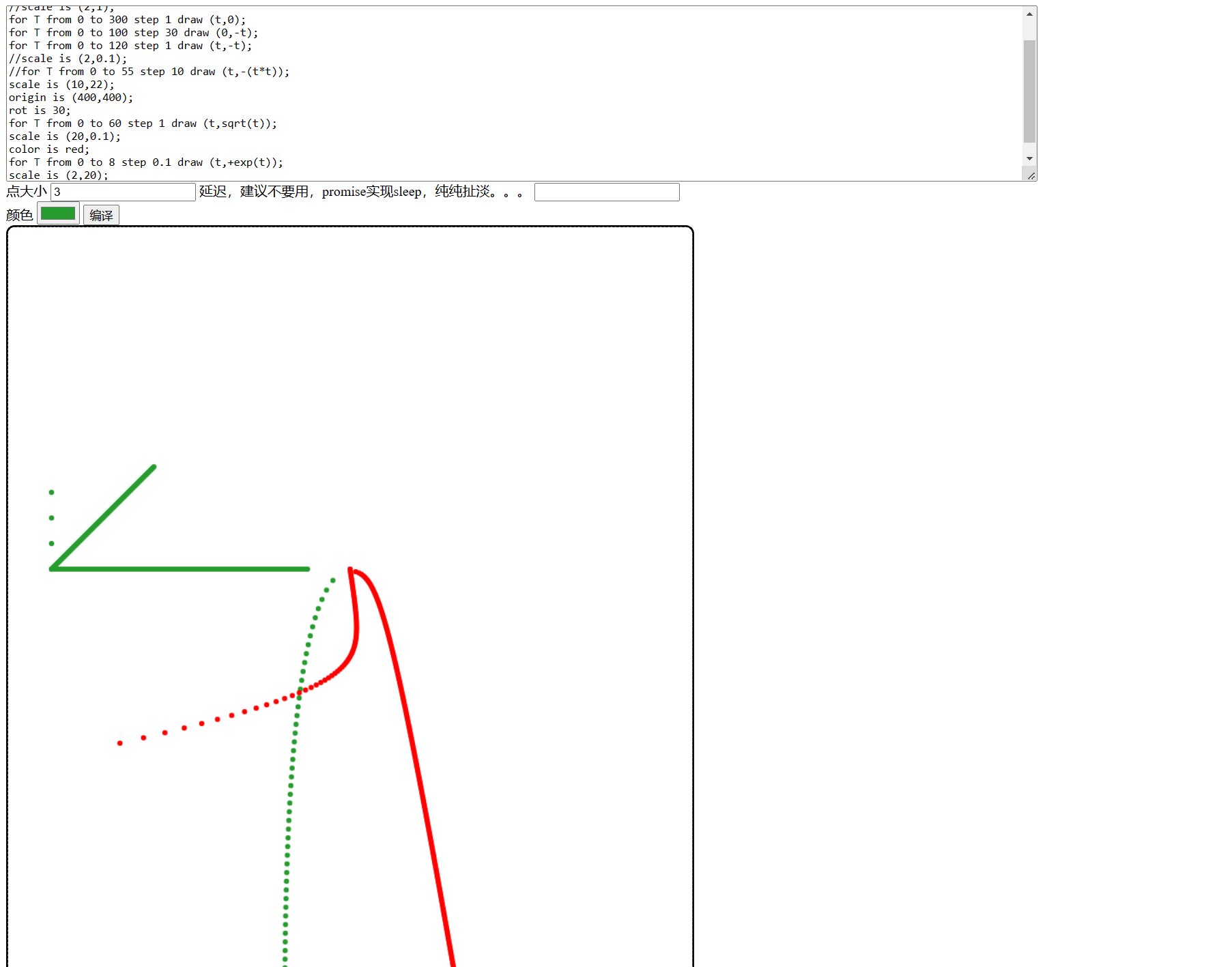
**一、实验目的**

1. 编写一个程序，对课件中要求的绘图语言进行编译并将正确的图像绘制到屏幕上。
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. 实现此程序的语法、词法、语义分析器。

**二、实验环境（软硬件环境）**

vsCode、配置好的webpack环境（编译ts到js）、chrome浏览器

**三、实验内容（设计/实现思路，即如何完成的。不允许贴大段代码）**

1. 问题分析
   1. 编程语言选择：虽然老师推荐使用java、c++这类面向对象的语言，但本人对于java、c++或者python的图形化知之甚少。并且作为一名前端爱好者，对于图形绘制这种任务首先想到的就是html中的canvas标签，canvas暴露了许多绘制API供开发者使用，然而此程序比较精密，使用弱类型的JS明显不太好，而作为强类型的js——ts则提供类、枚举、类型规范等一系列工具帮助开发者使用，同时也有面向对象的特性，因此我最终决定使用ts来完成此程序。
   2. 编译任务分析：乍一看此程序比较复杂，当时我们仔细分析可以得到一下内容。
      1. 就 5 种语句，而且很死板
         1. 设置源点
         2. 设置缩放
         3. 设置旋转
         4. 循环画图，唯一一个麻烦的
         5. 注释
         6. 加了个颜色设置
      2. 解释器，读一句做一句
      3. 词法分析器要干啥：读取输入的字符串，把能解析出来的东西变成定义好的 token 对象，传递给语法分析器使用。
      4. 语法分析器：总共就五条句子，直接硬编码匹配句型就行，唯一麻烦的就是表达式分析。
      5. 语义分析：真正往 canvas 上画画的东西，为了方便读一句做一句直接继承语法分析器。
2. 任务实现
   1. 目录结构及任务
   2. Scanner实现过程
      1. Scanner为词法分析器，首先我们在这里面定义了Token类，同时通过硬编码的方式在Token\_Type定义了所有会出现在程序中的Token类型，并在Token\_Table定义关键字。当然，对于可能出现的内置函数，在Func子目录中定义函数本体。而Scanner则是真正干活的程序，通过对传入的inputString逐个字符的阅读比对，产出Token。
      2. Scanner大致流程为：建立缓冲字符串，逐个阅读后将读到的字符存入缓冲字符串。首先必须进行预处理，即过滤程序中的空格、回车、TAB符等知道我们遇到有效的字符。其次开始读取字符，如果字符是字母，则此次Token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。
   3. Parser实现过程
      1. Parser作为语法分析器，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。在此目录我们定义了树节点类和变量类。
      2. Parser大致流程为：首先是阅读Token，根据“每句话”的第一个Token就可以知道这句话要干什么，Scale->放缩，Origin->源点设置，For->循环绘画。。。然后我们为每种语句对应一个函数来实现绘制。
      3. 如何让程序理解表达式：根据表达式的特点，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)。同时编写几个递归下降的函数（Term、Factor、Component、Atom、Expression）来实现结点的创建。匹配时按照表达式优先级自底向上建树，atom作为最底端负责创建常量、变量、函数结点，而只有函数可以返回Expression识别内容，Component负责识别乘方，Factor负责识别一元取正负，Term负责识别乘除，Expression负责识别加减。
   4. Semantic实现过程
      1. 语义分析器，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。
      2. Draw程序的作用：类似Semantic和Canvas的中间层，Semantic调用Draw的方法来与Canvas对话。
      3. Semantic：继承Parser，并且设置缩放、源点位置、旋转角度、变量信息，方便实现“读一句做一句”，当Parser完成工作，Semantic读取词法分析器得到的TreeNode并求值完成设置。比如源点设置，Parser完成工作后，x点和y点的根节点储存在x\_ptr和y\_ptr中，Semantic求得两个结点的具体值并进行设置。
      4. 如何求树结点值：按部就班，自底向上，遇到符号（+-\*/）进行计算，遇到函数调用函数体，遇到变量向语义分析器要变量的值。
      5. 如何绘制点：参考计算机图形学中的变换矩阵。
   5. 其他文件
      1. Index.html：提供网页的基本内容，如绘制需要的画布，提供输入程序本体功能的文本域，设置绘制点大小和颜色的input框。
      2. Index.ts：程序入口，负责清空画布并绘制与Semantic对话。
   6. 实验结果

**四、心得体会**

1. 优点
   1. 使用的并非传统的Java、Python等语言，而是ts这类比较小众却好使的语言。
   2. 并不是硬编码读文件来获取输入，而是提供给用户输入内容的能力，同时可以设置点大小和颜色。
2. 缺点
   1. 浏览器不像其他语言抛出异常直接停止允许，这会导致输入如果出现我没有预料的错误直接让Cpu占用达到100%（console.log问题）。
   2. 无法方便地实现延迟绘画功能，这是因为JS、浏览器的单线程（时间换问题）导致的，因为浏览器先同步、后异步，而如果希望实现延迟，不像Java、python直接sleep即可，需要引入Promise、async等一系列繁琐内容。
3. 总结
   1. 首先，虽然 ts 原生支持正则表达式，但是这个程序不用正则表达反而更简单
   2. 对我来说，这个大作业最好的收获就是复习了 ts，也方便了我后来用 tsx。
   3. 当然，最难的地方自然是递归下降分析程序，头脑风暴。
   4. 和以前读 Vue 源码的模板编译和生成的 ast 语法树不同，毕竟 html 自己就是树形结构。
   5. canvas 的实现绘画可能不大好用（要绘制点需要使用arc函数），其实一开始可以考虑用 Svg 绘制会更简单一些。