朱家辰 211250044

程序介绍

程序功能

1）识别输入程序存在的词法与语法错误
2）识别8、16进制整数与科学计数法的浮点数
3）在输入程序正确的前提下构建语法分析树

程序编译

在linux环境下，在Lab/code文件夹打开终端，
输入make命令可以生成可执行文件parser，
输入make test命令可以将test1.cmm作为输入程序开始进行分析，
输入./parser $输入程序文件$命令可以指定输入程序进行分析

实现方法

词法分析

给出规则的词法照抄即可，未给出规则的词法：
整数分类讨论为哪种进制类型，根据开头即可唯一确定，注意十进制数不能有前导0但其他进制都可以有
普通浮点数要求了小数点前后都有数字，科学计数法则只要求小数点两边有一边有数字即可（但在实际编译器实测中普通浮点数其实也可以写作类似.00的形式，这里仍旧按照定义）
在实际编译器中，以下两种一定被认定为数字：
1）以数字开头
2）以小数点开头且小数点后一位是数字
后面无论是数字还是字母都会被认定为数字，而如果满足上述对于数字的定义但不满足以上任何一种整数或浮点数的写法即认定为词法错误

语法分析

照抄给出的所有语法规则，将所有语法单元属性值定为一个结构体的指针，对于每个语法单元都新建一个对应结构体，结构体内存储所有语法单元构建语法树所需信息（定义在head.h中）。
对于错误恢复：
程序主要由一系列定义（函数或全局变量）组成，当某个定义无法继续匹配，则可以移入error认为当前定义完成匹配，尝试在剩下的内容中继续匹配下一个定义

函数、变量定义出错不再考虑其内包含的内容
在函数、变量体内，由一系列语句组成，当某个语句无法继续匹配，同理移入error尝试匹配下一条语句
具体实现即为添加$ExtDef \rightarrow error, Stmt \rightarrow error, Def \rightarrow error$三个产生式
因为进一步细分错误工作量太大，因此将错误简要分为以上三类进行恢复。
最后在主程序实现语法分析树的先序遍历即可

一个小问题

在实验过程中我发现了一个讲义存在的表述问题，通过自己的查证才得以解决，希望未来能修改一下讲义：
在2.2.11中，提到YY_USER_ACTION的宏定义确实可以直接写在Flex源文件中，
但%location这个选项应当写在Bison源文件中，Flex并不存在该选项，
写在Flex中的等价写法是%option bison-locations