编译原理 lab1 报告

请叫我 bug 小王子.

本次作业经历了非常痛苦的 debug 过程, 这种完全没有 bug 头绪的感觉还是第一次.

主要原因大概是, 这次的核心逻辑都是 antlr 实现的, 我自己实现的部分非常少以至于, 潜意识里就有一种 "怎么会错呢" 的感觉.

然而后来一个规则一个规则地 debug 时发现, 我果然是, 怎么写怎么有 (bug) 啊.

Implement

src 文件夹内容如下:

```
.
├── CmmLexer.g4
├── CmmLexerImpl.java
└── Main.java
```

其中 CmmLexer.g4 是词法单元. Main.main 是处理逻辑, 主要就是 getAllTokens 后判断有没有错误, 没有一个个输出, 有就直接退出.

getAllTokens 在扫描时已经把所有可能的词法错误报出来了.

CmmLexerImpl 是编译后得到的 CmmLexer 类的一个子类.

它按照要求重载了 notifyListeners 方法, 同时引入了 faults 变量来记录词法错误个数, 并且提供接口来访问它. faults 大于 0 说明有词法错, Main.main 就可以退出了.

Bug list

Fail to make

第一个遇到的 bug 是提交上去后 0 分. 这个困扰了我好长时间, 后来发现是本地 java 版本和云端 java 版本不一致, 云端不支持 var 所以报错.

我 lab0 没有特别认真做, 没有注意到要求用的是 open-jdk8... 直接用的我本机配置好的 open-jdk16.

Big decimal

然后是在检查 INT 时准备试一下上限和下限, 也就是 2^32 - 1. 一试果然就错了.

Java 的 Integer.parseInt 在遇见超过 2^31 - 1 的数时就会报错, 因为它是补码需要兼顾负数.

改用 BigInteger 类后问题解决.

Float Double

而后是浮点数的精度问题. 这个问题是通过 1.23e12 发现的, 它的输出是 1.22...

这个就是超过 32 位带符号浮点数的上界了, 改成 Double 解决.

0x(...)*

最后一块拼图是我今天早上心态稍微稳定了一点后 (没错, 我昨晚破防了), 重新一行一行读我的 g4 文件发现的.

我的 INT 的 8 进制和 16 进制的实现中, 对于数字序列部分用的是 '*' 而不是 '+', 这样在遇到单独的 '0x/0X' 时就会出错.