语法分析

上海交通大學

一、 目标

在之前的作业中，已经完成了 Seal++语言的词法分析器，词法分析提取出了 Seal++语言文件中的各个单词，本次作业的目标为：根据语法规则（Seal++语法手册中已给出，**其中语法形式定义已更新，请及时从 canvas 上下载**），使用 Bison 为 Seal++语言文件中识别出的单词生成一棵抽象语法树。

二、 Bison 简介

Bison 是一种常用的语法分析生成器，它基于 LALR 语法分析的原理，可以为我们自动构造一个语言分析器。其工作过程可以分为两个阶段：(1)根据 Seal++的语法说明，编写对应的 Bison 描述文件 **seal.y（这是此次作业要修改的文件）**，之后用 Bison 处理，生成对应的 c 文件。(2) 使用 c 编译器，进一步得到对应的语法处理程序 parser。

Bison 的描述文件由 3 部分组成：定义、规则和辅助程序，这三部分由顶行的两个连续百分号分割，具体如下：

```
定义部分
%%
规则部分
%%
辅助程序部分
```

各部分书写格式可以参考课本**附录 B 语法分析生成器 YACC、附件中的《flex 与 bison》或其他参考资料**。

对于课本中提到的 YACC，Bison 可以看作 YACC 的升级版，YACC 的规则在 Bison 中同样适用。

三、 文件说明

seal.y：此次作业要修改的文件。此文件给出的形式为一个代码框架，注释将会指明在何处添加代码，除了这部分之外，允许对其他部分、大体框架做任意的修改。此外，任何需要编写的额外的辅助函数，都应该包含在这个文件中。

此外，有若干的已经以 C++编写好的辅助包，这将大大减少同学们一些无关的繁杂的代码编写工作，**在修改 seal.y 时，同学们需要参考这些代码，具体说明见附带代码包.docx。**

四、 如何测试

准备：测试前保证机器上已安装 bison。安装**命令：sudo apt-get install bison**

编译：利用**命令 make parser** 可以编译对写好的代码进行编译，编译会产生 parser 文件，同时会产生 Seal.output 文件，里面包括了 LALR(1)解析表可以帮助调试例如移进规约冲突等问题。另外，请在每次生成分析器之前使用**命令 make clean** 清除临时文件，因为有时候代码的修改不能及时反映在临时文件中。

测试：利用**命令./parser testfile**，可以对名为 testfile 的文件进行检测。test 目录下给出了部分样例文件，对应的测试结果存放在 test-answer 目录下。可以利用自己编写的语

法分析器预先分析，然后与结果对比，结果要求与样例答案完全一致。对于有语法错误的样例文件，要求能够指出**错误位置**，但不必指出错误类型（只需要把规则写好，当检测到错误就会自动出现错误位置了）。

批量测试：使用**命令 sh judge.sh** 可批量对比 test 目录下的样例文件进行语法分析，输出结果会自动与 test-answer 目录下的参考答案进行比对，如果完全一致，屏幕上会显示“passed”。

五、 如何评分

作业提交之后，我们将根据同学提交的词法分析器对若干个已有的样本进行分析，其中样本可能包含语法错误，包含错误的样本要求输出错误及其行，给出出错位置通过，无错误样本要求将输出结果与标准结果比对，完全一致的通过，否则不通过。按照所有样本分析通过率给分（即例如满分 10 分，通过率 0%，给 0 分，通过率 70%，给 7 分）。

六、 文件提交要求

要求将原作业目录 syntex 下的所有文件，放置在一个名为<学号>的目录下，并且将整个目录打包为<学号>_<姓名>.tar 格式。

七、 提交截止时间

请同学们在 2021.12.19 晚上 23:59 之前，将结果提交到 canvas（将会查看文件创建时间，2021.12.19 晚上 23:59 之后的均视为迟交，结果按 0 分处理）。