语法分析

上海交通大學

一、 介绍

本次作业的目标为使用 bison 和一个用于处理 Seal 语法树的包为 Seal 编写一个解析器，即使用 bison 的语义操作一棵抽象语法树，该包在 Seal 语言操作手册给出。在此之前需要参考 Seal 的语法结构。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，本次作业将会对对应单词确定一个语法规则，根据定义的语法规则，单词最终应该能够归约到一棵语法树的树根。

二、 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，这些注释只是为了能够确保语法分析器能够正常工作。除了这部分之外，允许对其他部分、大体框架做任意的修改，如果需要帮助，请阅读 flex-bison 的手册。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。

- 测试文件

样例文件和测试结果分别在 test 目录及 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）。

三、 如何测试

利用 make parser 可以编译出自己的语法分析器文件，利用./parser testfile，可以对名为 testfile 的文件进行检测。在编译时，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。要求，**必须没有**冲突。

四、 输出

程序在 parser-phase.cc 的 main 函数里，调用了 ast_root 的 dump_with_types(cout,0) 函数，该函数利用递归的方式输出一棵抽象语法树（AST），AST 的根并且只有根为 program 类型，对于成功解析的程序，解析器的输出是 AST 的内容。 对于有错误的程序，输出为解析器的错误消息以及错误位置。我们提供了一个错误报告例程并且以标准格式打印错误消息。请不要对其进行修改。并且不应该在语义动作中直接调用此例程。当检测到问题时，bison 会自动调用它。另外，解析器只需要处理包含在单个文件中的程序，不必担心会编译多个文件。

五、 错误处理与恢复

语法解析器在分析过程中产生的任何错误，只需要指明错误出现的位置即可。

六、 树包

在 Seal 的附带代码包文档中，对 Seal 抽象语法树的代码包进行了说明，代码文件

随同作业一并给出。你将需要大部分这些信息来编写有效的解析器。

七、 操作提示

请大家不要忘记安装 bison。sudo apt-get install bison

现有的文档可能并不能解决同学们的所有问题，请同学们广泛查阅资料、代码，包括但不限于 stackoverflow、github 等。

八、 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）。

九、 文件提交要求

要求将原作业目录下的所有文件，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。

十、 提交截止时间

请同学们在 2020.11.29 晚上 23:59 之前，将结果提交到 ftp（将会查看文件创建时间，2020.11.29 晚上 23:59 之后的均视为迟交，结果按 0 分处理）。