编译原理实验三

20308003 曾伟超

EX1

EX1 部分的代码在 `ex1` 目录下，EX1 的部分主要为讨论 `Oberon-0` 这个实验语言的一些特性以及编写对应的测试样例，这部分的实验报告在 `ex1/Oberon-0.pdf`

EX2

EX2 部分的代码在 `ex2` 目录下，EX2 部分主要任务是使用一个类似于 `GNU Flex` 的词法分析器生成工具，来生成一个可供后续使用的一个词法分析工具，在这里为了方便 EX2 独立运行，在 EX2 的生成中加入了 `%standalone` 的选项，来生成一个内嵌的 `main` 函数，并可以通过 `./build.sh && ./run.sh <inputfile>` 的方式来运行测试用例，对应的测试结果会打印到标准输出中，这部分的实验报告在 `ex2/lexgen.pdf`

EX3

EX3 部分代码位于 `ex3` 目录下，这部分的主要任务是使用一个类似 `GNU Bison` / `YACC` 的一个语法分析器生成工具，来实现对 `Oberon-0` 的语法分析和语义分析，并在完成分析后通过实验软装置，来绘制一个函数调用图，实际运行可以使用 `./gen.sh && ./build.sh && ./run.sh <inputfile>` 来完成对 `<inputfile>` 的测试，对应的实验报告在 `ex3/yaccgen.pdf` 中

EX4

EX4 部分代码位于 `ex4` 目录中，这部分的任务为手写一个递归下降的 `Oberon-0` 的语法分析器，同时后续需要完成一个函数内部的流程图的分析，同时在实验报告中对比了在 `EX3` 部分使用的自底向上的语法分析器，两种语法分析技术之间的区别，这部分的实验报告在 `ex4/scheme.pdf` 中

实验心得

本次实验可以算是对编译原理这学期所有课程的一次小结，在理论课上所学的一个编译器的工作流程基本本次实验都走完了，除了后续的目标代码生成部分，这部分的工作，我认为可以通过 `AST` 的遍历来完成，通过一个递归的函数 `eval` 来完成，实际上，`ex4` 部分就是使用了这个思路，例如 `ex4/src/ast/stmt.java` 中的 `eval` ，这部分是用来生成流程图的，通过修改和增加部分辅助类，例如 `BasicBlock` 接口和其它的派生类，例如 `ifBlock` 之类的方法，类似 `ast` 的实现，可以将 `ast` 转换为 `basicblock` ，再通过 `BasicBlock` 遍历就可以生成线性的代码了，当然这部分还停留在想法的阶段，受限于期末阶段，大部分课程的期末大作业都集中在这段时间，所以时间原因，没有做进一步的扩展

但整体来说，本次实验对我来说还是收获颇丰的，从 0 开始实现了一个基本可以完成编译的编译器，让我对编译原理这门课有了更加深入的了解