编译原理实验报告 Lab1

郑伯霖 191240000 匡亚明学院 191240078@smail.nju.edu.cn

吴雨欣 191240060 匡亚明学院 191240060@smail.nju.edu.cn

1 项目环境

- GNU LINUX Release: Ubuntu 20.04.1
- GCC version: 9.4.0
- GNU FLEX version: 2.6.4
- GNU Bison version: 3.7.6

2 实现功能

2.1 词法分析

- 词法分析部分借助 GNU Flex 工具，通过编写词法单元对应的正则表达式及匹配动作生成扫描器。
- 当扫描器匹配到一个合法的词法单元时，会调用 **create_node** 创建一个终结符节点，用于语法树的构建。
- 非词法单元进行相应的报错。
- 能正确识别八进制、十六进制及浮点数；额外构造了错误数字输入的 TOKEN(如 0x6u96)，但仍返回相应的类型给 Bison，报错时会打印整串错误数。
- 注释内容滤除，不返回词法单元给 Bison；块注释嵌套块注释、块注释嵌套行注释均正确报错。

2.2 语法分析

- 语法分析部分借助 GNU Bison 工具，通过编写文法生成式和对应动作生成分析器
- 当分析器识别到一个文法时，会构建一个非终结字符节点，并调用 **add_child** 进行父子节点的连接

2.3 语法树

- 设计了以 CST_node 为结点的多叉树结构。总体上分两大类结点，非终结符结点与终结符结点。终结符结点又可细分为五小类。从而用不同类型的结点存储不同类型的属性值信息。
- CST 结构中定义的 compact_type 由 SYM_TYPE 和 NODE_TYPE 构成，用于储存额外信息。前者代表该节点存储的符号的信息，后者表明该节点的类型。
- 在全局变量 ExtDef，语句 Stmt 以及表达式 Exp 层次实现了错误恢复功能，并使用 Bison 的%destructor 动作释放抛弃的符号属性值所分配得的内存避免泄露。
- 外部接口主要有:
  - struct CST_node * creat_node();
  - bool add_child();

  - void destory_node();
  - void destory_tree();
  - struct CST_node * copy_node();

  用于 Flex 和 Bison 中的节点构建以及连接操作。

3 编译方法

- 利用提供的 Makefile 文件，使用 make 进行编译。在 Code 目录下键入 make 即可完成编译。
- 执行
  - flex lexical.l
  - bison -d suntax.y
  - gcc main.c syntax.tab.c error.c -lfl -o parser

  进行编译，然后通过./parser test.cmm 进行测试。