哈尔滨工业大学计算学部

实验报告

课程名称：编　　译　　原　　理

课程类型：　　限　　选

实验题目：语　　义　　分　　析

学号：1190201019

姓名：罗家乐

一、 功能及实现

实验二在实验一的基础上，实现了语法树的解析，符号表的构建，与实验要求的语义错误提示。

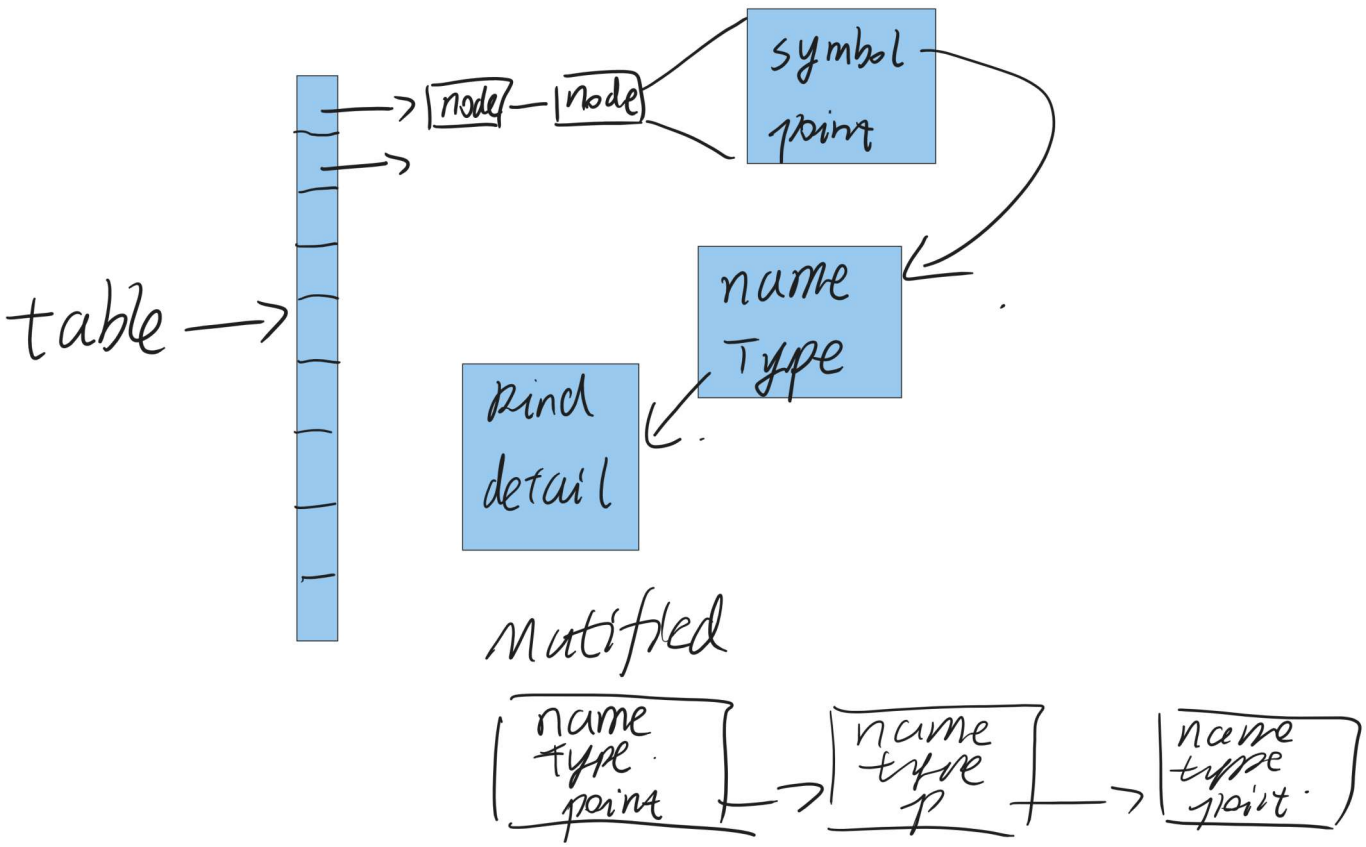
语法树解析

按照先序遍历法，对实验一中生成的语法生成树进行先序遍历。其中，对于不同非终结符调用不同函数，以进行不同类型的处理。

符号表构建

要进行后续的语义检查与及可能的中间代码生成，还需要收集代码中定义的结构、变量，以及函数的参数和返回值，为此，符号表的构建是必须的。

使用如下的数据结构对解析出的符号信息进行存储。



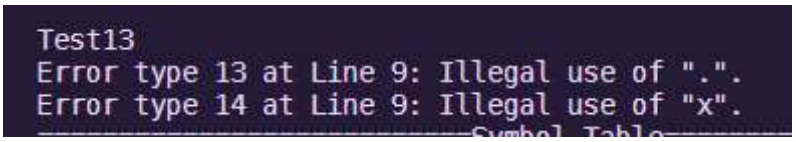
生成的符号表如下。

```
=========================Symbol Table=========================
1. Name: p Type: struct Position{Name:x Type: float at line 3; Name:y Type: float at line 3; } at line 8;
2. Name: struct Position Type: struct Position{Name:x Type: float at line 3; Name:y Type: float at line 3; } at line 1;
3. Name: main Type: function ()->int at line 6;
=============================End==============================
```

语义错误提示

借由在解析语义的过程中参照已生成的符号表，即可进行重复定义、缺少定义、类型不匹配等语义错误的提示。

语义提示示例：



二、 编译、使用及测试方式

本次实验文件提供两种测试方式：

手工测试

使用文件附带的makefile，通过make指令，获取最终程序parser。再使用测试用例，逐一测试。

```
make
./parser test/test1.c //test1.c test2.c test3.c …… test17.c
```

shell脚本自动测试

使用编写好的Test.sh脚本自动进行测试。

```
./Test.sh
```

Test脚本将自动编译源文件生成parser，执行17个测试用例，然后make clean消除生成的文件。

parser将输出语义提示错误以及生成的符号表。