实验二

Chenning Yu 151242062 匡亚明学院 151242062@smail.nju.edu.cn

程序应该如何被编译?

- `make all` 会依次按照 Syntax/syntax.y 生成 Syntax/syntax.tab.c, Syntax/syntax.tab.h, 按照 Lex/lexical.l 生成 Lex/lex.yy.c，最后生成与 Makefile 处于同一文件夹下的 parser。该 parser 能够用来测试实验指南中出现的样例，并按情况生成相关的语义错误。
- `make debug` 是为了协助我 debug 语法生成过程的命令。它与 `make all` 生成 parser 的过程一致，不过这里的 parser 会开启诊断模式，即：运行时会输出状态机的转移过程。
- `make testSeman1`，`make testSeman2`，`……`，`make testSeman17` 分别对应于实验指南中的必做样例 1 到样例 17。
- `make testSeman18`，`make testSeman19`，`……`，`make testSeman23` 分别对应于实验指南中的选做样例 1 到样例 6。

对于助教测试，建议先敲入 `make all` 重新生成一遍 parser。

如需测试已有样例，则使用上述的 `make testSemanN` 命令； `N` 为对应的样例编号。

如需测试新的样例，请使用 `./parser path/to/testfile` 命令； `path/to/testfile` 对应新的样例的相对路径。

本次实验的主要代码位于 `Semantics` 文件夹中。

实验实现了哪些功能?

完成了实验二的必做与选做部分的所有内容。粗体字为本代码亮点。

- 必做的错误类型
    - 我生成一个如实验指南中的十字链表+哈希表：

在每次进/出 CompSt 结构体的时候（它最左为 `{`，最右为 `}`），在栈中压入/弹出一个符号链表的表头，并把相应的符号在哈希表中生成/删除。

- 我对于每一种产生式进行细心的枚举，并把它们根据父节点的类型放在不同的函数中。
- 写着写着，我封装出了一些函数，因为它们的重复率很高。比如检查两个类型之间是否相同，以及输出不同序号的错误。

- 选做的错误类型
  - 要求 2.1: 修改前面的 C-- 语言假设 3，使其变为「函数除了在定义之外还可以进行声明」。

    解：我在 lexical.y 中加入新的产生式：`ExtDef → Specifier FunDec SEMI`. 在哈希表里，插入新函数时，检查该此函数的来源是声明还是定义。此外，我在程序最后检查哈希表中是否还剩下未被定义的函数。通过以上方法，我通过了测试样例。
  - 要求 2.2: 支持变量嵌套定义。

    解：通过上面的哈希&十字链表可轻松扩展。
  - 要求 2.3: 从名等价变为结构等价。

    解：对于两个结构体之间的类型检查，顺序地比较结构体中域的类型。若比较结果是均为一致，则结构等价。否则不等价。

- **亮点**：
  - **有针对不同的错误恢复的不同的提示。**如针对选做样例2，parser会输出：

    ```
    Error type 19 at Line 8: "func": Inconsistent declaration of function.
    Error type 18 at Line 6: "func": Undefined function.
    ```
  - **恢复之后的语法树可以进行输出。**此功能需要使用 `./parser path/to/testfile -debug` 命令，可以看见程序恢复错误语法之后生成的语法树。

实验总结

两个疑问与我自己的解法

疑问1

问：如果某函数要求返回 `INT` 值，但在函数体内并没有 `RETURN` 语句。虽然实验指南中并没有要求这类错误的检查，语义错误分析器检查得出来吗?

答：我写的 parser 检查不出来。考虑到 `IF-ELSE` 语句，若要实现这个功能，分析器需要在每一个条件分支中检查是否有 RETURN 语句，这可能会带来性能上的降低。此外，对于 `WHILE LP Exp RP Stmt` 语句，若条件 `Exp` 恒等于 1，程序还需要检查 `Stmt` 中是否存在 `RETURN` 语句。

如果需要扩展，使语义分析器检查所有分支的话，就在每个产生式所在函数中新加一个布尔值，代表该产生式是否在一个函数体中。若该布尔值为真，则检查该产生式是否有 `RETURN` 语句；如有新的分支，则递归检查子产生式是否有 `RETURN` 语句。

疑问2

问：有两个结构体如下，他们是类型等价吗?

```
struct s1{                          struct s2 {
    int a;                              char b;
    char b;                             char c;
    char c;                             int a;
};                                  }
```

答：我的 parser 会认为这两个结构体不等价，因为我的 parser 是通过顺序检查对应次序的域的类型是否相符，而第一个域的类型就不相符。

如果需要扩展，使这两个乱序的结构体等价的话，个人认为就不能够使用结构等价了。比如需要针对类型一中的每个域的名字，查找类型二中是否存在对应的名字，并检查这一对域的类型是否相等。

小结

- 简单的剑法组合起来能够屠龙。一开始，我基本上写的都是比较短的函数，比如建立哈希表。原因是我不知道复杂的该怎么写。不过，慢慢地我在面临复杂的产生式时，发现可以递归调用那些已经写过的简单的函数。于是我的剑法虽然没有长进，不过组合渐渐变多、变花哨了。最后完成了十九个语义错误的输出，我也写了近八百行。其中大部分行都是在赋值、以及调用其他的函数，没做什么高深的算法。对我来说，这些代码可读性也很好。这使我感受到了简单的魅力。
- 通过这次实验，我接触了符号表的建立，上下文无关文法，结构体类型的表示等概念，对于 C-- 每个产生式都有了深刻的了解，甚至可能能背出来。此外，进一步完成了函数声明、变量嵌套定义、结构等价等需求。此外，也创造了本学期一周内写过代码最多的纪录。收获颇多。