编译原理 实验二报告

181860155 朱晓晴

181860155@smail.nju.edu.cn

1 编译方式

Linux 版本：Ubuntu 16.04（32 位）

在/Lab/Code 目录下执行以下指令进行编译，得到可执行文件 parser：

```
make parser
```

执行以下指令进行测试，path 为测试文件路径：

```
./parser <path>
```

2 程序功能

2.1 数据结构

考虑到本次实验需要实现选做要求 2.2，实验选用基于链表的 open hashing 散列表的 Imperative Style 的符号表。符号表的结构与实验手册中所提及的基本一致，不同之处有：

- 为 structure 类型添加 tail 域，便于定义 struct 时插入新定义的域；
- 为符号添加名称来源信息，包括函数名、结构名、全局变量名和局部变量名，便于插入和使用符号时对符号表进行查询。

2.2 错误处理

基于实验手册和部分自行定义，本次实验对于变量重名的判断如下表所示：（√表示允许同名，×表示不允许同名，黑色表示实验手册定义，红色表示自定义）

名称来源	函数	结构体	全局变量	局部变量/形参	域
函数	×	√	√	√	√
结构体	√	×	×	×	√
全局变量	√	×	×	√	×
局部变量/形参	√	×	√	×（同一语句块） √（不同语句块）	√
域	√	√	×	√	×

除函数之间、结构体之间、全局变量之间和同一语句块的局部变量（形参）之间重名之外，其他情况下的符号重名都选择报错，但仍将所有重名符号尽可能存入符号表中，使得编译器尽量少报源于同一根本语义错误的错误。由于符号表中存放了符号来源信息，因此在使用符号时可以根据来源获取相应的符号，符号表中存在额外的重名不影响编译器正常运行。

此外，如果查询符号表后没有得到相应的符号，编译器会返回特定的错误符号errorSymbol。同时，如果处理 Exp 时出现符号未定义或类型不匹配等错误，会返回错误类型errorSpecifier。后续处理时，如果识别到 errorSpecifier，编译器会允许它与其他任意数据类型匹配，以避免同一行连续报相同的类型不匹配的错误。

3 心得体会

1. 由于 Lab2 涉及较多对指针的使用，最初测试时会出现较多的段错误，而简单地向 stdout 或 stderr 打印错误信息并不实用，基本无法在段错误发生前输出有效信息。因此，后续采用了修改 makefile 并使用 gdb 进行调试的方法，基本能够精准快速地找出 bug。在进行了一定量的测试后，段错误较少发生时，选择在 stdout 输出手册要求的报错，在 stderr 输出 debug 信息（包括非终结符处理进度、打印符号表和符号栈等），并将 stderr 重定向到 log 文件。在部分测试用例中，尽管报错正确，但通过观察 log 文件可以发现符号表维护过程中出现的错误。
2. C--语言对重名的限制相较于 C 语言更为严格，例如不同结构体的域不能重名、结构体和变量不能重名。如果处理重名时直接丢弃最新的重名符号，不另加处理，就会出现一连串相同原因的报错。尽管重名限制降低了代码编写的难度，但会降低 C--语言的实用性。
3. 编译器目前以递归形式处理 xxxList，在处理多维数组时性能一般，后续需要进行优化。