编译原理实验二

211220028 党任飞

一、实现功能

- 对于有词法、语法错误的程序，正确报错。
- 对于存在手册中提及的17种语义错误的程序，正确报错。
- 实现针对BASIC、ARRAY、STRUCTURE三种类型变量的嵌套作用域，外层语句块中定义的变量可在内层语句块中重复定义，内层语句块中定义的变量到了外层语句块中就会消亡。

二、运行方式

进入 `Lab2/Code/` 目录之后，执行

```
> make
> ./parser test.cmm
```

即可编译生成 `parser` ，并对任意 `test.cmm` 文件进行分析。

三、核心实现方法

在上一次实验的基础上，增加
了 `symbolTable.c` 、 `symbolTable.h` 、 `semanticAnalysis.c` 、 `semanticAnalysis.h` 四个文件。其中：

- 前两个文件负责实现符号表。
  - 实现了 `Type_` 、 `FieldList_` 两个表示类别和域的结构
  - 定义了 `TableNode_` 结构，用以实现符号表
    - `TableNode SymbolTable[TABLE_SIZE];`
  - 用另外一个数组（栈）记录十字链表，实现嵌套作用域。维护一个 `depth` 变量记录嵌套深度，同时作为栈顶指针。设置了 `MAX_DEPTH=50` ，希望测试用例没有比这个还大的。
    - `TableNode CrossTable[MAX_DEPTH];`
    - 在退出一个嵌套作用域的时候，执行 `void leaveScope(int depth)` 函数，删除 `CrossTable` 中的对应链接关系，但不改变全局符号表的结构。
- 后两个文件负责进行分析。

- 给每一个文法符号单独写一个check函数，按语法树结构递归进行检查。

四、声明

我的实现逻辑是遇到错误就报告，然后把当前分析的这一个元素类型设为 `NULL` ，而 `NULL` 在与任何类型计算相等与否的时候都会报错，因此一旦出现出错的内容，后续一定会连带报出类型不匹配的错误。例如：

- 重复定义变量：不会把这个变量加入符号表（防止冲突）
  - 与作用域内变量重名，则后续使用该变量的任何地方都可能会报出类型不匹配5/7等错误
  - 与结构体重名，则后续使用时报出变量未定义错误1
- 使用未声明的变量，则一定会连带报出类型不匹配5/7等错误
- 重复定义函数，则在调用的时候只认可定义的第一个函数，如果按照后面的函数形参传参则报错9
- 类似的其他错误

个人认为我报出了“本质错误”，故不对此特性进行修改。