编译原理实验报告

实验名称：实验一 词法分析与语法分析

指导教师：许畅

实验时间：2023.3.11

任务号：14

报告人：

分配	姓名	学号
组长	张铭俊	201220065
组员	吴浩然	201220064

一、程序功能

必做内容

- 你的程序需要对输入文件进行语义分析（输入文件中可能包含函数、结构体、一维和高维数组）并检查类型1-17的错误。

选做内容

- 修改前面的C−−语言假设5，将结构体间的类型等价机制由名等价改为结构等价（Structural Equivalence）。例如，虽然名称不同，但两个结构体类型struct a { int x; float y; }和struct b { int y; float z; }仍然是等价的类型。注意，在结构等价时不要将数组展开来判断，例如struct A { int a; struct { float f; int i; } b[10]; }和struct B { struct { int i; float f; } b[10]; int b;}是不等价的。在新的假设5下，完成错误类型1至17的检查。

二、程序运行

1. 程序划分

程序主要划分为了以下几个文件

- lexical.l：词法分析文件，用于给出对应词法的正则表达式和匹配时的动作，能够生成lex.yy.c
- syntax.y: 语法分析文件，用于给出语法的产生式以及产生式推导时的对应动作，能够生成syntax.tab.h及syntax.tab.c
- main.c: 主函数
- tree.c: 存放全局变量如firstNode(语法分析树头节点)，语法分析树节点结构体Node，此次语义分析新增的四个符号表（变量，函数，数组，结构体）等等

2. 程序实现

1. 四个符号表

   此次符号的类型type采用pdf所给的Type_与FieldList进行实现，再次不再进行展示，通过线性链表（虽然普通但确实好用！）实现了四个符号表（变量、函数、数组、结构体），对于每一个符号表，有固定的new...()和find...()来对符号进行创建和查询，在创建函数的时候，我引入了额外的函数getdetype()来获取函数形参，和getretype()获取形参，checktype()来判断是否参数数量类型满足要求，再次仅展示函数与数组的符号链表节点定义

```
typedef struct tArray_
{
    char *name;
    Type type;        // 类型
    char *type_name; // 返回值名称
    int isStruct;     // 是否为结构体域
    int structNum;    // 如果是结构体域属于第几号结构体
    int structIndex; // 如果是结构体域中的是第几个结构体
    struct tArray_ *next;
} Array_;

// 函数符号表
typedef struct tFunc_
{
    char *name;
    Type returnType;  // 返回值类型
    char *type_name;  // 返回值名称
    int varNum;       // 形参个数
    Type varType[10]; // 参数类型，参数个数不超过10个
    struct tFunc_ *next;
} Func_;
```

2. 判断变量嵌套

在判断变量嵌套方面，考虑到可能出现结构体嵌套结构体，嵌套的结构体属于不同层因而变量可以重名情况，我引入了栈来进行操作，栈顶层指向目前所在的结构体域的结构体号，来指明变量所在域，这一点贯彻我的程序始终，不论是在判断是否变量重名，结构体是否可以找到变量都有着用处。

```
// 存放结构体域，目前假设结构体嵌套不超过10层
int structNumStack[10];
// 栈顶
int stackTop;
// 目前结构体总数量
int maxStructNum;
// 目前变量数量
int structIndexStack[10];
int stackIndexTop;
// 结构体名称所在的行
int structopttagline[10];
```

3. 实现选作结构体等价

在这一模块，由于此前设计数据结构的合理性，我仅需书写两个额外函数isSameTypeByNode(tNode *,tNode *)与isSameType(Type t1,Type t2)即可以完成判断，通过对两个type的kind进行判断，进而转而对type的u进行遍历判断，因而很顺利的便执行成功了选做2-3。

```
// 根据Type判断结点的类型是否相同,相同返回0,不同返回1
int isSameTypeByNode(tNode *n1, tNode *n2);
int isSameType(Type t1, Type t2);
```

4. 判断返回值类型

由于考虑到返回值类型可能同一个函数对应多个，而函数不会出现多个嵌套定义情况，因此我引入了一个Type型数组和返回值数量来记录目前函数的return后的返回值，来对specifier中明确给出的返回值类型进行比较，若有一个不一致，就报一次错误8。

```
for (int i = 0; i < returnTypeNum; i++)
{
     if (isSameType(funcTail->returnType, funcReturnType[i]) != 0) // 返回值类型不相同，报8号错误
     {
          printf("Error type 8 at Line %d: Type mismatched for return.\n", returnlineno[i]);
     }
}
```

5. 编译方式

编译的三条指令如下

```
bison -d syntax.y
flex lexical.l
gcc syntax.tab.c main.c tree.c -lfl -ly -o parser
```

其余部分应该与大部分同学完成的并无太大差别，因而再次不再赘述。