1.文件结构

- lexical.l
- main.c
- Makefile
- syntax.y
- tree.c
- tree.h
- hashtable.h
- hashtable.c//关于哈希表与栈的一系列操作
- semantic.h
- semantic.c//语义分析识别的操作

2.数据结构

c对应的参数类型

```
    struct Type_
{
    Kind kind;
    union
{
// 基本类型
int basic;
// 数组类型信息包括元素类型与数组大小构成
struct { pType elem; int size; } array;
// 结构体类型信息是一个链表
FieldList structure;
//函数类型
struct{pType returntype;FieldList args;} function;
} u;
};
struct FieldList_
{
    char* name; // 域的名字
    pType type; // 域的类型
    FieldList tail; // 下一个域
};
```

底层结构：采用十字链表构成的哈希表与栈结合的符号表结构

```
typedef struct hashTable_{
    item* hashlist;
}HashTable;
HashTable table;
HashTable functiontable;
HashTable declaretable;
typedef struct stack_{
    item* stacklist;
int stacknum;
}Stack;
Stack stack;
```

其中table存放变量与结构体信息 functiontable提供函数信息 declaretable存放声明函数信息

对于每一个哈希的信息存放

```
struct item_{
int line;//行数
int value;//是否被定义
int depth;//栈的定义深度
FieldList field;
item nextHash;
};
```

3.处理过程

```
    table =inithashtable();
    functiontable =inithashtable();
    declaretable =inithashtable();//初始化定义函数表.变量定义表.函数表
    stack = initstack();//初始化栈
    visit(root);调用处理函数，对每一个节点分析
```

```
    visittable();//识别未定义的声明函数
```

4.错误处理

统一使用printerror（）函数进行处理，传入一定的参数数据 因为大部分返回值都是一个指针，报错后，统一返回空指针，相当于返回了一个统一的报错指针，在对应的处理中空指针不做处理。

5.实验感想

在完成要求时，需要对已经完成的代码进行对应的修改，需要对整体有很好的了解才能进行处理。在对应的地方的增删，需要深刻理解。