【lab2】实验报告

一、实现的基本功能。

实现语义分析使用的是L属性文法，就是只要前面的信息收集完了，我就结合这些信息来进行插表和查表。完成的选秀是2.2，我采用的符号表组织方式是：栈+链表。

在具体传递继承属性的过程中，我使用的是传递一个Type数据类型，里面可以表示变量、函数、结构体定义和结构体的变量，同时采用了递归的思想，因为函数的返回值、参数很有可能也是结构体、变量等等，于是里面再加上Type *来递归表示。

```
struct mytype
{
  int type;  // function or variable or array 数据类型
  int val_type; //0 means int, 1 means float [ val: value ,array:element's type]
  char* id_name; //变量名称
  int * array_dims; //本次实验没用到
  int array_size; //数组的维数
  union val
  {
    int int_val;
    float float_val;
  }value;
  int line; // 行号

  int is_parameter; //analyze whether a parameter
  int func_paras_num; //函数参数的个数
  struct mytype  *paras; //函数的参数，是一个Type类型的链表
  struct mytype  *return_type; //函数的返回值也是一个Type*类型

  //STRUCT
  char * struct_name; //结构体的名字，因为只用实现名等价
  Fieldlist * head; //结构体的域
};
```

举一个赋值语句的例子,来说明如何语义分析：

```
_Exp(root->child[0],before);
_Exp(root->child[2],after);
whether_same_type(before,after);
```

很简单，就是用_Exp函数对等号左右分别做语义分析，分别报底层的错误，分析完后信息都保存在了before和after指针里面，然后我在用一个whether_same_type函数检查两者的类型是否匹配，检查更高级一点的语义错误就可以了。

二、如何使用我的程序。

在Code文件夹里面输入：

```
make clean
make
```

即可得到parser文件，用parser文件来检测cmm文件

三、个人亮点。

1、符号表如何添加struct的定义。

我并没有使用Type类型去收集struct里面的域，相反，我的做法是：因为我实现的是2.2，所以我将struct当作一个函数，它定义的时候，我必须向符号表的栈里面压入新的符号表，这就给我了一个途径，没有必要通过传递Type指针来获取域，而是在要退出struct的时候遍历当前符号表，将其添加到struct里面就行了，减少了参数的传递量。下面是具体的代码：

```
    //向符号表里面压入一个新的表
    Add_symbol_table(_type);
    /*
    这个是为了报错不同，因为struct里面重复定义叫redefiend field 而正常时redefined
variable所以这里标识一下，
    好处理报错。
    */
    set_in_struct();
    _DefList(root->child[3]);//进入域里面
    set_out_struct();
    //在Delete_current_table删除当前符号表之前将参数都传进struct对应的Type
    Symbol * new_struct=create_struct_symbol(_type);
    //get out of field
    Delete_current_table();//删除当前符号表
    //add symbol
    insert_table(new_struct);//将struct的定义加入上一个域中
```