编译原理实验三实验报告

161220092 孟华 161220092@smail.nju.edu.cn

161220091 马可欣 3476799672@qq.com

一、实现功能

- 在词法分析、语法分析和语义分析程序的基础上，将C--源代码翻译为中间代码，并输出为线性结构，通过提供的虚拟机小程序测试运行结果。
- 除了基本要求外，对选做3.1进行了实现，支持结构体变量作为函数参数。

二、如何编译

- 编译：make
- 运行：./parser test out.ir
- 测试：python irsim/irsim.pyc

三、实验数据结构

因为完成实验二的时候，轻信了一句话“语法树之后都没有用了”， 所以将语法树所有的相关实现都删掉了，事实证明.....orz

好的，所以实验三既要用到实验一的语法树又要用到实验二的生成符号表的相关操作，而为了最大程度地减少代码的更改量，我们最终的解决方案是，将两个实验的数据结构结合起来，封装成一个更高层次的结构体作为所有Variable的type。具体如下：

```
struct Node_ { //存放语法树的节点
    char *name;
    enum NodeType nodeType;
    char stringValue[300];
    int numValue;
    float numValueF;
    int childNum;
    Node child[8];
};
```

```
struct FieldList_ //存放实验二中变量的节点
{
    char name[maxIdLength];
    Type type;
    FieldList next;
    char isLeftValue[10];
    char isFollowEqual[10];
```

```
    int line;
    enum { DEFVAR, DECVAR } status;
    enum { NPAVAR, PAVAR } var;//实验三新增，判断结构体是否为函数参数
};

struct FuncList_ //存放实验二中函数的节点
{
    enum { DEF, DEC } status;
    Type return_type;
    char name[maxIdLength];
    FieldList parameters;
    FuncList next;
    int line;
};
```

```
struct SyntaxFieldNode_ //结合实验一实验二的结构体
{
    FieldList field;
    Node node;
};

struct SyntaxFuncNode_
{
    FuncList func;
    Node node;
};
```

```
typedef struct YYSTYPE //更新后的YYSTYPE
{
    int type_int;
    float type_float;
    SyntaxCharNode type_char;
    SyntaxTypeNode type_type;
    SyntaxFieldNode type_field;
    SyntaxFuncNode type_func;
} YYSTYPE;
```

这样就可以完美地将实验一实验二的结构体融合到一起，生成语法树的同时建立符号表等。

因为一开始也思考过边建立语法树边生成中间代码，但是后来发现有一些综合属性的参数值并不能传递下去，而且规约方向往往与生成中间代码方向相反，遂放弃， 改为建好语法树后对于语法树进行递归翻译。

四、问题及思考

- 结构体既可作为函数参数也可以作为变量申明，作为函数参数时，因为传入的是地址，所以可以直接用取值（t2 ： = * v1）；但当结构体作为函数体内变量时，需要先对结构体取地址（t9 ： = &v3），

再进行取值，所以两种情况下翻译的中间代码时不同的。针对这个问题，我们在FieldList结构体中新增了一个变量，用来判断结构体是否为函数参数，以区分中间代码。

- 传递实参时，需要维护一个arg_list数组，我们定义了一个全局的arg_list数组，但当出现多个带参数的函数时，原来的arg_list会被覆盖，导致原来的函数参数无法正确打印，这是因为我们在生成ARGUMENT类型的操作数时，传入的是一个FieldList类型的地址，所以在对其进行直接赋值时，操作数指向的是一个地址，因此里面的值会被之后新的arg_list覆盖。因此在生成ARGUMENT类型的操作数时，我们申请了一个新的FieldList内存空间，再进行赋值，此时前参数的值便不会被后一个参数值覆盖。
- 在函数定义时递归会出现函数未定义的错误，因此我们先将报错信息存储，函数规约完成后再判断是否报错。