lab3实验报告

语义分析中，去掉了实验二检查错误的内容，只保留了构建符号表的内容

与实验二类似，采用SDT，对于每个语法树的节点（每个语法单元），单独用一个翻译函数进行处理，

先参考讲义中最基本的结构表达式写翻译模板，再拓展到其他表达式

选做实现了结构体的翻译：需要计算要找的首地址

数据结构：

操作数Operand：

```
struct Operand_ {
    enum {      //kind of operand
        VARIABLE,       // v1
        TEMP,           // t2
        PARAMETER,      // v3
        CONSTANT,       // #4
        LAB,            // label5
        FUNCT           // foo
    } kind;
    // is value or address (for assign)
    enum { VAL,
        ADDRESS } type;
    // moreinfo
    union {
        int var_no;         // nomber for v,t,p,label
        long long int value; // value for c
        char* func_name;    // function name for f
    } u;
};
```

单行中间代码：

```
struct InterCode {
    enum {  //kind of intercode
        LABEL,  // LABEL x:
        FUNCTION,   // FUNCTION f:
        ASSIGN,
                // check operand is value or address in type below
        ADD,        // x := y + z
        SUB,        // x := y - z
        MUL,        // x := y * z
        DIV,        // x := y / z
        GOTO,       // GOTO x
        IF,         // IF x [relop] y GOTO z
        RETURN,     // RETURN x
        DEC,        // DEC x [size]
        ARG,        // ARG x
        CALL,       // x := CALL f
        PARAM,      // PARAM x
        READ,       // READ x
        WRITE       // WRITE x
    } kind;
    enum {      // type for assign
        NORMAL,     // x := y
        GETADDR,    // x := &y
        GETVAL,     // x := *y
        SETVAL,     // *x := y
        COPY        // *x := *y
    } type;
    union {
        struct {    // for assign
            Operand *left, *right;  // left := right
        } assign;
        struct {    // binary operand for +-*/
            Operand *res, *op1, *op2;   // res := op1 op op2
        } binop;
        struct {    // single operand for call
            Operand *res, *op;  // res = CALL op
        } sinop;
        struct {    // for read,write,label,goto,return...
            Operand* op;    // READ op
        } single;
```

```
        struct {    // conditional for if
            Operand *op1, *op2, *target; // IF op1 [relop] op2 GOTO
target
            char relop[4];
        } cond;
        struct {    // declearation for DEC
            Operand* op;    // DEC op size
            unsigned size;
        } dec;
    } u;
};
```

operand这里用type为assign语句特殊判断左值和右值是value or address，并为intercode设置type为普通赋值NORMAL/取内容赋值GETADDR/赋值地址SETVAL/..

全部中间代码（双向链表）：

```
struct InterCodes {
    InterCode code;
    bool isDelete;
    InterCodes *prev, *next;
};
```