姓名：白文强 学号：20191060064 专业：计算机科学与技术

**一、实验目的**

1. 掌握单遍扫描编译程序的组织方法；
2. 掌握符号表的基本管理方法；
3. 掌握中间代码生成的一般方法。

**二、实验内容**

完成词法、语法、语义分析的编译器前端。具体要求包括：

1. **实现算术运算+、-、\*、/运算，运算满足左结合，且满足\*，/优先级别高于+、-；将合法的输入翻译成三地址代码；**
2. **类型检查：即对所有的运算包括赋值，需要考虑运算对象的数据类型是否一致，不一致报告语义错；**
3. **禁止同名标识符的重复申明；**
4. **实验分析与设计**

（1）根据文法以及语义，得到如下所示的翻译模式：

start→ DS.  
D→ B; D  
D→ε  
B→ int L **{ L.type := int }** | real L **{ L.type := real }**  
L→ id **{ A.type := L.type enter(v.entry, L.type)}** A  
A→ ,idA **{ A1.type := A.type enter(v.entry,A.type)}**  
A→ε  
S→ V := E **{ gen( ":=", E.place,0,V.place) }** H **{if E.type != H.type error\_type; else V.type=E.type}**  
H→;S | ε  
E→T **{ R.place:=T.place}** R **{E.place:=R.place; if T.type != R.type error\_type; else E.type=T.type}**  
R→+T **{ R1.place:= newtemp; gen( "+", R.place, T.place , R1.place) }** R **{R.place=R1.place; R.type=R1.type}**  
R→-T **{ R1.place:= newtemp; gen( "-", R.place, T.place , R1.place) }** R **{R.place:=R1.place; R.type=R1.type}**  
R→ ε **{R.place=R.place; R.type=R.type}**  
T→F **{ P.place:=F.place}** P **{ T.place:=P.place; if F.type != P.type error\_type; else T.type=F.type}**  
P→\* F **{ P1.place:= newtemp; gen( "\*", P.place, F.place ,P1.place) }** P**{P.place:= P1.place;P.type=P1.type }**  
P—>/ F **{ P1.i:= newtemp; gen( "/", P.place, F.place , P1.place) }** P **{P.place:= P1.place;P.type=P1.type }**  
P→ε **{P.place=P.place; P.type=P.type}**  
F→( E ) **{ F.place := E.place; F.type=E.type}**  
F→id **{F.place := found(id).place; F.type=found(id).type}**  
V→id **{V.place:=found(id).place; V.type=found(id).type}**

（2）程序结构说明

typedef struct info

{

    enum symbol type; *// 类型*

    int place;*//*

} info;

info结构体，用于存放变量的类型以及位置信息

void init(); *// 初始化函数*

int getsym(); *//读取下一个单词，获取其类型*

void enter(enum symbol type);*//在符号表中插入当前变量*

info found(char \*idt);*//查找变量在符号表中的位置和类型，没有则返回0*

int gen(enum symbol op, int arg1, int arg2, int result); *//中间代码分析*

void writecode(char \*op, char \*arg1, char \*arg2, char \*result); *//写缓存*

void start(); *//分析入口*

void D(); *// 对定义变量语句进行分析*

void B(); *// 对定义变量语句进行分析*

void L(enum symbol type); *//对定义变量语句进行分析*

void A(enum symbol type); *//对定义变量语句进行分析*

void S(); *//分析赋值运算*

void H(); *//分析下一条运算语句*

info E(); *//分析赋值号后面的运算语句*

info R(info Ri); *//检验加减法运算*

info T(); *//检验加减法运算*

info P(info Pi); *//检验乘除法运算*

info F(); *//检验乘除法运算*

info V(); *//对变量进行分析*

（2）功能实现与分析

在原有的基础上，实现了：

1. 加减乘除**运算，运算满足左结合，且满足乘除优先级别高于加减；**

**增加乘除运算，添加函数P(info Pi)，分别处理乘除运算，此处仅放乘法的代码。**

if (sym == times)

{

    getsym();

    F\_info = F();

    if (Pi.type == F\_info.type)

    {

        tv = tv + 1;

        gen(times, Pi.place, F\_info.place, tv);

        P\_info = P(F\_info);

    }

    else

    {

        printf("语义错，运算类型不一致");

        exit(0);

    }

}

1. **类型检查，包括赋值类型检查以及运算类型检查；**

**为了实现赋值类型检查，在S()函数中，得到V()返回值中的类型信息以及E()返回值中的类型信息，将其进行比较，如果二者类型不一致，则说明出现了类型赋值类型不一致的情况，进行报错。**

V\_info = V();

if (sym == becomes) *//当前单词为:=*

{

    getsym();

    E\_info = E();

    if (V\_info.type == E\_info.type)

    {

        gen(becomes, E\_info.place, -1, V\_info.place);

    }

    else

    {

        printf("语义错，赋值两端类型不一致");

        exit(0);

    }

    H();

}

else

{

    printf("语法错，缺少等于号");

    exit(0);

}

**为了实现运算类型检查，在P()、R()、E()、T()函数中，也增加了类型判断，如在T()中，由于T->FP，则需要将F以及P的类型进行检查**

F\_info = F();

P\_info = P(F\_info);

if (P\_info.type == F\_info.type)

{

    T\_info.type = P\_info.type;

    T\_info.place = P\_info.place;

}

else

{

    printf("语义错，运算类型不一致");

    exit(0);

}

1. **重复定义检查**

为了进行重复定义检查，在插入符号表时，需要进行查询，如果在符号表中出现过当前变量，则说明出现了重复定义。

for (int i = 1; i < tx; i++)

{

    if (strcmp(table[i].name, id) == 0)

    {

        printf("重复定义: %s\n", id);

        exit(0);

    }

}

1. **实现与结果分析**

对文件t1.txt:

real a,b;

int c,d;

c:=c+d;

a:=b\*a.



对文件t2.txt

real a,b;

int c,d;

c:=c+d;

a:=b\*a+c.



对文件t3.txt

real a,b;

int c,d;

c:=c+d;

d:=b\*a.



对文件t4.txt

real a,b;

int c,b;

a:=a+b.



对文件t5.txt

real a,b;

int c,d;

c:=c+d;

b:=e\*a.



从上面的结果中可以看到，程序可以对正确程序做出正确的返回，能够对运算类型错误、赋值两端类型不一致错误、重复定义错误、变量未声明错误等进行正确的检查，完成了实验要求，符合语法规则。