**语义检查与中间代码生成实验报告**

**一、实验目的**

通过设计、开发一个高级语言的语义检查和中间代码生成程序，加深对相关课堂教学内容，包括语法制导翻译技术、类型确定、类型检查、常见可执行语句（如赋值语句、条件语句、循环语句）翻译技术的理解。

**二、实验要求**

(1)了解语义检查和中间代码生成的目的和意义；

(2)掌握语义检查和中间代码生成的一般内容；

(3)掌握语法制导翻译技术（特别是针对S-属性定义的自底向上翻

译方法和针对L-属性定义的深度优先翻译方法）；

(4)掌握根据翻译目标编制语义子程序的方法（可模仿教材中的语义规则进行设计）。

**三、实验原理**

语义分析是任何编译程序必不可少的一个阶段， 在整个编译过程中，词法分析和语法分析是对源程序形式上的识别和处理，而语义分析程序是对源程序的语义做相应的处理工作。中间代码生成不是编译器的必须阶段，生成中间代码的目的是为了便于优化和移植。

语义分析和中间代码生成的主流技术是语法制导翻译技术。 语法制导翻译的基本思想： 为每个产生式配上一个语义子程序，（该子程序描述了一个产生式所对应的翻译工作。这些工作包括：生成中间代码，查填有关的符号表，检查和报错，修改编译程序某些工作变量的值等）。在语法分析过程中，每当一个产生式用于推导（自顶向下分析）或归约（自底向上分析）时，就调用该产生式所对应的语义子程序，以完成既定的翻译任务。

**四、实验步骤**

(1)熟悉S语言的语义，了解语义分析阶段需要进行哪些语义检查和需要对哪些可执行语句进行翻译；

(2)分析S语言的文法，必要的话可对文法进行改写（如增加标记非终结符号并为其添加产生式）；

(3)为相应产生式编写语义子程序（模仿教材中的语义规则进行设计），实现对运算及运算分量进行类型检查（运算的合法性与运算分量类型的一致性或相容性）、变量定义的唯一性检查等语义检查工作；

(4)理解掌握变量的中间代码、表达式的中间代码、语句的中间代码的结构（见教材）；

(5)编写语义子程序（模仿教材中的语义规则进行设计），实现S语言中各类语句（赋值语句、条件语句、循环语句）的翻译工作，将源程序翻译为中间代码。

(6)撰写实验报告。

**四、实验报告**

**1.算法实现原理**

●算术表达式文法：

G(E)： E 🡪 E ω0 T | T

T 🡪 T ω1 F | F

F 🡪 i | (E)

●文法变换：

G’(E) E 🡪 T {ω0 T}

T 🡪 F {ω1 F}

F 🡪 i | (E)

●属性翻译文法：

E 🡪 T {ω0 “push(SYN, w)” T “QUAT”}

T 🡪 F {ω1 “push(SYN, w)” F “QUAT”}

F 🡪 i “push(SEM, entry(w))” | (E)

其中：

push(SYN, w) — 当前单词w入算符栈SYN;

push(SEM, entry(w)) — 当前w在符号表中的入口值压入语义栈SEM;

QUAT — 生成四元式函数

i．T = newtemp;

ii．QT[j] =( SYN[k], SEM[s-1], SEM[s], T); j++;

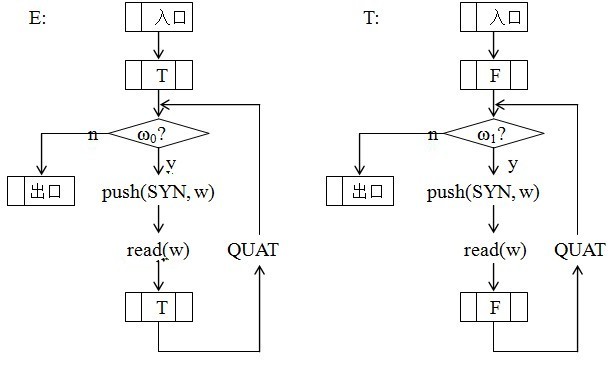
iii．pop( SYN, \_ ); pop( SEM, \_ ); pop( SEM, \_ );

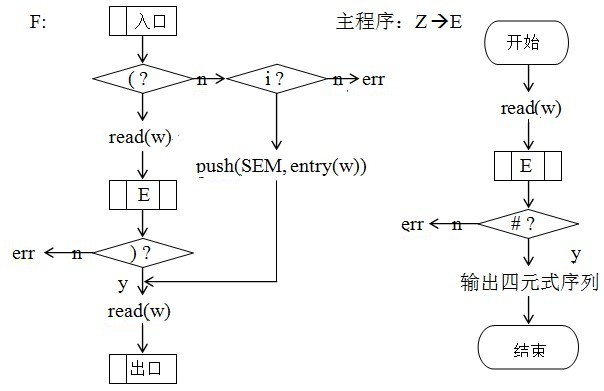
push( SEM, T );

●递归下降子程序：

数据结构：SYN —算符栈；

SEM —语义栈；





**2.数据结构设计**

使用递归的结构进行四元式的设计，同时，运用堆栈结构将四元式的输出序列打印出来：

while ( exp[i]=='+' || exp[i]=='-'){

syn[++i\_syn]=exp[i]; //push(SYN,w)

i++; //read(w)

T();

quat();}

while ( exp[i]=='\*' || exp[i]=='/'){

syn[++i\_syn]=exp[i]; //push(SYN,w)

i++; //read(w)

F();

quat();}

void quat(){

strcpy(qt[j],"(, , , )"); //QT[j]:=(SYN[k],SEM[s-1],SEM[s],temp);

qt[j][1]=syn[i\_syn];

qt[j][3]=sem[i\_sem-1];

qt[j][5]=sem[i\_sem];

qt[j][7]=temp;

j++;

i\_syn--; //pop(SYN);

i\_sem--; //pop(SEM);

i\_sem--; //pop(SEM);

sem[++i\_sem]=temp; //push(SEM,temp);

temp++;}

**3. 中间代码生成程序代码（使用递归下降语法分析）**

#include<stdio.h>

#include<string.h>

#include<iostream>

using namespace std;

void Print();

void Scanner();

bool Statement();

bool Condition();

bool While\_Statement();

bool Expression();

bool Item\_expression();

bool Factor();

bool Conditional\_statements();

bool Assignment\_statement();

bool Compound\_statements();

char ch;

int sum;

char token[10];

char program[300] = " \

const x=8,y=7; \

Var a, b; \

begin \

    a = x + y; \

    b = a\*x; \

end#";

char \*rwtab[9] = { "begin","if","then",

"else","while","do","const"

,"var","end" };

int m;

int syn;

int p;

int t = 1;

char word[10];

char var[10];

bool isDigital(char ch)

{

    if (ch <= '9'&&ch >= '0')

        return true;

    else

        return false;

}

bool isAlpha(char ch)

{

    if (ch >= 'a'&&ch <= 'z' || ch >= 'A'&&ch <= 'Z')

        return true;

    else

        return false;

}

void Scanner()

{

    int num = 0;

    for (m = 0; m<10; m++)

    {

        token[m] = '\0';

    }

    m = 0;

    ch = program[p++];

    while (ch == ' ' || ch == '\n' || ch == '\t')

    {

        ch = program[p++];

    }

    if (isAlpha(ch)) {

        do {

            token[m++] = ch;

            ch = program[p++];

        } while (isAlpha(ch) || isDigital(ch));

        p--;

        syn = 10;

        token[m++] = '\0';

        for (int n = 0; n <= 8; n++)

        {

            if (strcmp(token, rwtab[n]) == 0)

            {

                syn = n + 1;

                break;

            }

        }

        return;

    }

    else if (isDigital(ch))

    {

        sum = 0;

        while (isDigital(ch))

        {

            sum = sum \* 10 + ch - '0';

            ch = program[p++];

        }

        p--;

        syn = 11;

        if (isAlpha(ch))

            syn = -1;

        return;

    }

    else

    {

        token[0] = ch;

        switch (ch)

        {

        case '<':ch = program[p++];

            if (ch == '>')

            {

                syn = 22;

                token[1] = ch;

            }

            else if (ch == '=')

            {

                syn = 18;

                token[1] = ch;

            }

            else {

                syn = 19;

                p--;

            }

            break;

        case '>':ch = program[p++];

            if (ch == '=')

            {

                syn = 21;

                token[1] = ch;

            }

            else {

                syn = 20;

                p--;

            }

            break;

        case '=':ch = program[p++];

            if (ch == '=')

            {

                syn = 17;

                token[1] = ch;

            }

            else {

                syn = 16;

                p--;

            }

            break;

        case '+':syn = 12; break;

        case '-':syn = 13; break;

        case '\*':syn = 14; break;

        case '/':syn = 15; break;

        case ';':syn = 23; break;

        case '(':syn = 24; break;

        case ')':syn = 25; break;

        case ',':syn = 26; break;

        case '#':syn = 0; break;

        default:syn = -1; break;

        }

        return;

    }

}

bool Constan\_Defined()

{

    if (syn == 10)

    {

        //cout << "常量定义" << token << endl;

        Scanner();

        if (syn == 16)

        {

            //cout << "等于" << token << endl;

            Scanner();

            if (syn == 11)

            {

                //cout << "无符号整数" << sum << endl;

                return true;

            }

            return false;

        }

        return false;

    }

    return false;

}

bool Constan\_Description()

{

    Scanner();

    if (syn == 7)

    {

        //cout << "常量说明" << token << endl;

        Scanner();

        while (Constan\_Defined())

        {

            Scanner();

            if (syn == 23)

            {

                //cout << "分号" << token << endl;

                return true;

            }

            else if (syn == 26)

            {

                //cout << "逗号" << token << endl;

                Scanner();

                continue;

            }

            //cout << "常量说明错误" << endl;

        }

    }

    else

        return false;

}

bool Variable\_Defined()

{

    if (syn == 10)

    {

        //cout << "变量定义" << token << endl;

        return true;

    }

    else

        return false;

}

bool Variable\_Description()

{

    if (syn == 10 || syn == 2 || syn == 5 || syn == 1 || syn == 0)

    {

        return true;

    }

    Scanner();

    if (syn == 8)

    {

        //cout << "变量说明" << token << endl;

        Scanner();

        while (Variable\_Defined())

        {

            Scanner();

            if (syn == 23)

            {

                //cout << "分号" << token << endl;

                return true;

            }

            else if (syn == 26)

            {

                //cout << "逗号" << token << endl;

                Scanner();

                continue;

            }

        }

    }

}

bool Condition()

{

    //cout << "条件" << endl;

    Expression();

    if (syn == 17 || syn == 18 || syn == 19 || syn == 20 || syn == 21

        || syn == 22)

    {

        //cout << "关系运算符" << token << endl;

        Scanner();

    }

    else

    {

        //cout << "关系运算符错误" << endl;

        return false;

    }

    Expression();

}

bool Expression()

{

    //cout << "表达式" << endl;

    //Scanner();

    do {

        if (syn == 12 || syn == 13)

        {

            //cout << "加法运算符" << token << endl;

            cout << "+";

            Scanner();

            Item\_expression();

        }

        else

        {

            Item\_expression();

        }

    } while (syn == 12 || syn == 13);

    return true;

}

bool Item\_expression()

{

    //cout << "项" << endl;

    while (Factor())

    {

        //Scanner();

        if (syn == 14 || syn == 15)

        {

            //cout << "乘法运算符" << token << endl;

            cout << "\*";

            Scanner();

        }

        else {

            return true;

        }

    }

    return false;

}

bool Factor()

{

    //cout << "因子" << endl;

    //Scanner();

    if (syn == 10)

    {

        //cout << "标识符" << token << endl;

        cout << token;

        Scanner();//特殊

        return true;

    }

    else if (syn == 11)

    {

        cout << sum;

        //cout << "无符号数字" << sum << endl;

        Scanner();

        return true;

    }

    else if (syn == 24)

    {

        //cout << "左括号" << token << endl;

        Scanner();

        Expression();

        if (syn == 25)

        {

            //cout << "右括号" << token << endl;

            Scanner();

            return true;

        }

        else

        {

            //cout << "没有），错误" << endl;

            return false;

        }

    }

    else

    {

        //cout << "没有左括号" << endl;

        return false;

    }

    return false;

}

bool Assignment\_statement()

{

    //cout << "赋值语句" << endl;

    //cout << "标识符" << token << endl;

    strcpy(word, token);

    Scanner();

    if (syn == 16)

    {

        //cout << "赋值语句=" << endl;

        Scanner();

    //  cout << endl;

        cout << "t" << t << "=";

        Expression();

    //  cout << endl;

        return true;

    }

    else

    {

        //cout << "没有=" << endl;

    }

}

bool Compound\_statements()

{

    //cout << "复合语句" << token << endl;

    Scanner();

    while (Statement())

    {

        if (syn == 23)

        {

            //cout << "复合语句中的分割符" << token << endl;

            cout << endl;

            cout << word << "=" << "t" << t << endl;

            t++;

            Scanner();

            if (syn == 9) {

                //cout<<"复合语句"<<token<<endl;

                break;

            }

        }

    }

    if (syn == 9)

    {

        //cout << "复合语句" << token << endl;

        Scanner();

        return true;

    }

    else

    {

        //cout << "复合语句缺乏" << endl;

        return false;

    }

    if (syn == 0)

    {

        //cout << "复合语句" << token << endl;

    }

}

bool Conditional\_statements()

{

    if (syn == 2)

    {

        //cout << "条件语句if" << endl;

        Scanner();

        Condition();

        if (syn == 3)

        {

            //cout << "then" << endl;

            Scanner();

            Statement();

            if (syn == 4)

            {

                Scanner();

                Statement();

            }

            else

            {

                return true;

            }

        }

        else

        {

            //cout << "条件语句中缺少 then" << endl;

            return false;

        }

    }

    else

    {

        return false;

    }

}

bool While\_Statement()

{

    //cout << "循环语句" << token << endl;

    Scanner();

    Condition();

    if (syn == 6)

    {

        //cout << "while循环的do" << endl;

        Scanner();

        Statement();

        return true;

    }

    // else

    // return false;

    else

        return false;

}

bool Statement()

{

    if (syn == 10)

    {

        Assignment\_statement();

        return true;

    }

    else if (syn == 5)

    {

        While\_Statement();

        return true;

    }

    else if (syn == 1)

    {

        Compound\_statements();

        return true;

    }

    else if (syn == 2)

    {

        Conditional\_statements();

        return true;

    }

    else

    {

        return false;

    }

}

int main()

{

    freopen("result.txt", "w", stdout);

    printf("中间代码生成：\n");

    p = 0;

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "程序开始" << endl;

    Constan\_Description();

    Variable\_Description();

    do {

        Scanner();

        Statement();

    } while (syn != 0);

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

    cout << "程序结束" << endl;

    cout << "按任意键退出" << endl;

}

**输入数据包含在实现代码中，以字符串的形式表示：**

char program[300] = " \

const x=8,y=7; \

Var a, b; \

begin \

    a = x + y; \

    b = a\*x; \

end#";

**输出结果如下：**

中间代码生成：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

程序开始

t1=x+y

a=t1

t2=a\*x

b=t2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

程序结束

按任意键退出