**递归下降语法分析实验报告**

**一、实验目的**

通过设计、开发一个高级语言的递归下降语法分析程序，实现对词法分析程序所提供的单词序列进行语法检查和结构分析，加深对相关课堂教学内容的理解，提高语法分析方法的实践能力。

**二、实验要求**

(1)理解语法分析在编译程序中的作用，以及它与词法分析程序的

关系；

(2)掌握递归下降语法分析方法的主要原理；

(3)理解递归下降分析法对文法的要求；

(4)熟练掌握Select集合的求解方法；

(5)熟练掌握文法变换方法（消除左递归和提取公因子）。

**三、实验原理**

(1)根据S语言BNF形式的语法规则（见附件A），写出S语言的上下

文无关文法；

(2)求每个产生式的Select集：

Select(AÆβ)= First(β)，当 ε∉First(β)

= (First(β)-{ε})∪Follow(A)，当ε∈First(β)

(3)判断是否满足递归下降法分析条件，若不满足用消除左递归和

提取公因子等文法等价变换操作对文法进行变换，使其满足递

归下降法的要求；

(4)构造递归下降语法分析程序，对文法中的每个非终结符号按其

产生式结构产生相应的语法分析子程序,完成相应的识别任

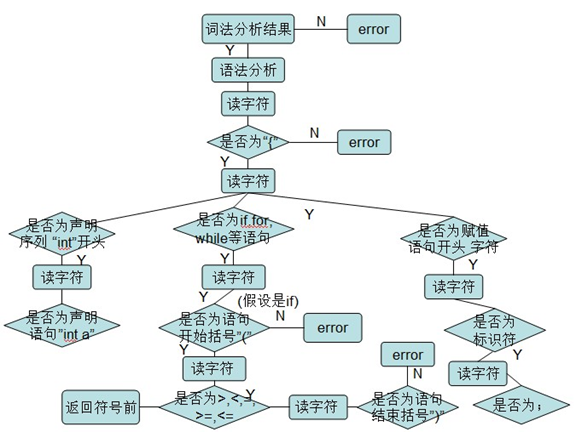
务。其中终结符号产生匹配命令，非终结符号则产生调用命令。

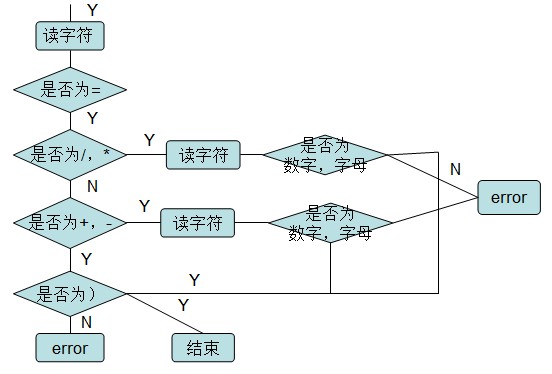
实际的语法分析工作从调用主程序（开始符号S对应的程序）

开始，根据产生式递归调用各个分析子程序；

(5)撰写实验报告

1. **实验报告**

**1.词法分析流程图如下**

对于字符的详细判断如下：

**2.递归下降语法分析程序代码**

#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();

int syn;//存放单词的类型

int p;

int numof\_ = 0;

char ch;

int sum;//用来保存数字的值

char token[10];

char program[300] = " \

Const x=8,y=7; \

Var a, b, c; \

begin \

a = b + x; \

    if a > 0 \

    then    \

        begin \

         c = y - 1; \

         a = a + 2; \

    end \

    while a>0 \

   do a = a - 1; \

end#";

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

"else","while","do","Const"

,"Var","end" };

int m;

void printNumOf() {

    for (int i = 0; i < numof\_; ++i)

        cout << " ";

}

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)

    {

        printNumOf();

        cout << "常量定义" << endl;

        numof\_ += 4;

        printNumOf();

        cout << token << endl;

        Scanner();

        if (syn == 16)

        {

            printNumOf();

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

            Scanner();

            if (syn == 11)

            {

                printNumOf();

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

                numof\_ -= 4;

                return true;

            }

            return false;

        }

        return false;

    }

    return false;

}

bool Constan\_Description()

{

    Scanner();

    if (syn == 7)

    {

        printNumOf();

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

        numof\_ += 4;

        Scanner();

        while (Constan\_Defined())

        {

            Scanner();

            if (syn == 23)

            {

                printNumOf();

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

                numof\_ -= 4;

                return true;

            }

            else if (syn == 26)

            {

                printNumOf();

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

                Scanner();

                continue;

            }

            cout << "常量说明错误" << endl;

        }

    }

    else

        return false;

}

bool Variable\_Defined()

{

    if (syn == 10)

    {

        printNumOf();

        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)

    {

        printNumOf();

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

        numof\_ += 4;

        Scanner();

        while (Variable\_Defined())

        {

            Scanner();

            if (syn == 23)

            {

                printNumOf();

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

                numof\_ -= 4;

                return true;

            }

            else if (syn == 26)

            {

                printNumOf();

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

                Scanner();

                continue;

            }

        }

    }

}

bool Condition()

{

    printNumOf();

    cout << "条件" << endl;

    Expression();

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

        || syn == 22)

    {

        printNumOf();

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

        Scanner();

    }

    else

    {

        printNumOf();

        cout << "关系运算符错误" << endl;

        return false;

    }

    Expression();

}

bool Expression()

{

    printNumOf();

    cout << "表达式" << endl;

    numof\_ += 4;

    do {

        if (syn == 12 || syn == 13)

        {

            printNumOf();

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

            Scanner();

            Item\_expression();

        }

        else

        {

            Item\_expression();

        }

    } while (syn == 12 || syn == 13);

    numof\_ -= 4;

    return true;

}

bool Item\_expression()

{

    printNumOf();

    cout << "项" << endl;

    numof\_ += 4;

    while (Factor())

    {

        //Scanner();

        if (syn == 14 || syn == 15)

        {

            printNumOf();

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

            Scanner();

        }

        else {

            numof\_ -= 4;

            return true;

        }

    }

    numof\_ -= 4;

    return false;

}

bool Factor()

{

    printNumOf();

    cout << "因子" << endl;

    numof\_ += 4;

    if (syn == 10)

    {

        printNumOf();

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

        Scanner();

        numof\_ -= 4;

        return true;

    }

    else if (syn == 11)

    {

        printNumOf();

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

        Scanner();

        numof\_ -= 4;

        return true;

    }

    else if (syn == 24)

    {

        printNumOf();

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

        Scanner();

        Expression();

        if (syn == 25)

        {

            printNumOf();

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

            Scanner();

            numof\_ -= 4;

            return true;

        }

        else

        {

            cout << "没有），错误" << endl;

            numof\_ -= 4;

            return false;

        }

    }

    else

    {

        cout << "没有左括号" << endl;

        numof\_ -= 4;

        return false;

    }

    numof\_ -= 4;

    return false;

}

bool Assignment\_statement()

{

    printNumOf();

    cout << "赋值语句" << endl;

    numof\_ += 4;

    printNumOf();

    cout << "标识符" << token << endl;

    Scanner();

    if (syn == 16)

    {

        printNumOf();

        cout << "赋值语句=" << endl;

        numof\_ += 4;

        Scanner();

        Expression();

        numof\_ -= 8;

        return true;

    }

    else

    {

        cout << "没有=" << endl;

    }

}

bool Compound\_statements()

{

    printNumOf();

    cout << "复合语句" << token << endl;

    numof\_ += 4;

    Scanner();

    while (Statement())

    {

        if (syn == 23)

        {

            printNumOf();

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

            Scanner();

            if (syn == 9) {

                break;

            }

        }

    }

    if (syn == 9)

    {

        numof\_ -= 4;

        printNumOf();

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

        Scanner();

        return true;

    }

    else

    {

        cout << "复合语句缺乏" << endl;

        return false;

    }

    if (syn == 0)

    {

        printNumOf();

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

    }

}

bool Conditional\_statements()

{

    if (syn == 2)

    {

        printNumOf();

        cout << "条件语句if" << endl;

        Scanner();

        Condition();

        if (syn == 3)

        {

            printNumOf();

            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()

{

    printNumOf();

    cout << "循环语句" << token << endl;

    Scanner();

    Condition();

    if (syn == 6)

    {

        printNumOf();

        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("语法分析：\n");

    p = 0;

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

    cout << "程序开始" << endl;

    numof\_ += 4;

    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;

    return 0;

}

**输入已经以字符串的形式包含在实现函数的主体中，如此处的输入函数为：**

char program[300] = " \

Const x=8,y=7; \

Var a, b, c; \

begin \

a = b + x; \

    if a > 0 \

    then    \

        begin \

         c = y - 1; \

         a = a + 2; \

    end \

    while a>0 \

   do a = a - 1; \

end#";

**输出结果为：**

语法分析：

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

程序开始

常量说明Const

常量定义

x

等于=

无符号整数8

逗号,

常量定义

y

等于=

无符号整数7

分号;

变量说明Var

变量定义a

逗号,

变量定义b

逗号,

变量定义c

分号;

复合语句begin

赋值语句

标识符a

赋值语句=

表达式

项

因子

标识符b

加法运算符+

项

因子

标识符x

复合语句中的分割符;

条件语句if

条件

表达式

项

因子

标识符a

关系运算符>

表达式

项

因子

无符号数字0

then

复合语句begin

赋值语句

标识符c

赋值语句=

表达式

项

因子

标识符y

加法运算符-

项

因子

无符号数字1

复合语句中的分割符;

赋值语句

标识符a

赋值语句=

表达式

项

因子

标识符a

加法运算符+

项

因子

无符号数字2

复合语句中的分割符;

复合语句end

循环语句while

条件

表达式

项

因子

标识符a

关系运算符>

表达式

项

因子

无符号数字0

while循环的do

赋值语句

标识符a

赋值语句=

表达式

项

因子

标识符a

加法运算符-

项

因子

无符号数字1

复合语句中的分割符;

复合语句end

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

程序结束

按任意键退出