

鄂尔多斯应用技术学院实验报告



ORDOS INSTITUTE OF TECHNOLOGY

Experiment Report

实验科目 (Subject)	<u>编译原理</u>	系别 (Dept.)	<u>数学与计算机工程系</u>
年级 (Grade)	<u>20 级计科一班</u>	专业 (Major)	<u>计算机科学与技术</u>
姓名 (Name)	<u>陈腾飞</u>	学号 (NO.)	<u>20207110036</u>
组别 (Group)	<u></u>	任课教师 (Teacher)	<u>王海军</u>

鄂尔多斯应用技术学院

实验分数：

实验项目	占比	得分
词法分析器的设计	25%	
语法分析器的设计	25%	
语义分析器的设计	25%	
综合实验	25%	
总分	100%	

书写规范要求：

1. 实验目的：清晰的说明本实验的目的和主要实验方法。
2. 实验内容与步骤：较详细地描述实验过程，并对数据、实验结果进行分析。
3. 实验心得：完成实验后自己的一些想法和体会，以及自己对实验方法的理解。

评分标准：

（一）A（85 分以上）：

1. 报告中对实验过程叙述详细、概念正确，语言表达准确，结构严谨，条理清楚，逻辑性强，自己努力完成，没有抄袭。
2. 对实验过程中存在问题分析详细透彻、规范、全面；结合企业资源战略方面内容描述正确、深刻。
3. 实验心得体会深刻、有创意，论述合理详细，有自己的个人见解和想法，能结合实例论述，提出问题并给出解决方法。

（二）B（75-85 分）：

1. 报告中对实验过程叙述较详细、概念正确，语言表达准确，结构严谨，条理清楚，逻辑性强，自己努力完成，没有抄袭。
2. 对实验过程中存在问题分析详细透彻、规范、全面；能结合实例内容描述正确。
3. 实验心得体会深刻、有创意，论述合理详细，有自己的个人见解和想法。

（三）C（60-75 分）：

1. 报告中对实验过程叙述较详细，自己努力完成，没有抄袭。
2. 对实验过程中存在问题有简单分析和描述，不全面。
3. 实验心得体会不够深刻，缺乏创意。

(四) D (60 分以下，或具备下面一项者为不及格)：

1. 没有交报告。
2. 基本上是抄袭。
3. 内容太空泛，太简单。

实验名称	实验一 词法分析器的设计
目的要求	设计一个简单的词法分析器，从而进一步加深对词法分析器工作原理的理解。 本实验的重点是理解词法分析器的输入与输出；难点是专用符号的识别。
实验内容	<p>1、该词法分析器要求至少能够识别以下几类单词：</p> <p>（1）关键字：else if int return void while 共 6 个，所有的关键字都是保留字，并且必须是小写；</p> <p>（2）标识符：识别与 C 语言词法规定相一致的标识符，通过下列正则表达式定义：ID = letter (letter digit)*；</p> <p>（3）常数：NUM=(+ - ε)digit digit*(.digit digit* ε)(e(+ - ε) digit digit* ε)，letter = a .. z A .. Z ，digit = 0 .. 9，包括整数，如 123，-123，+123 等；小数，如 123.45，+123.45，-123.45；科学计数法表示的常数，如+1.23e3，-2.3e-9；</p> <p>（4）专用符号：+ - * / < <= > >= != = ; , () [] { } /* */。</p> <p>2、分析器的输入为由上述几类单词构成的程序，输出为该段程序的机内表示形式，即关键字、运算符、界限符变为其对应的机内符，常数使用二进制形式，标识符使用相应的标识符表指针表示。</p> <p>3、词法分析器应当能够指出源程序中的词法错误，如不可识别的符号、错误的词法等。</p>
实验时数	4 学时
实验步骤	<p>(1) 定义模拟的简单语言的词法构成，调试词法分析程序；</p> <p>(2) 要求将用模拟语言书写的源程序进行词法分析；</p> <p>(3) 输出源程序运行结果，若分析有错误，须输出错误内容。</p>

实验过程

1、实验代码

```
import java.io.File;
import java.io.FileReader;

/**
 * 此程序是通过将文件的字符读取到字符数组中去，然后遍历数组，将读取的字符进行
 * 分类并输出
 * @author
 *
 */
```

```

    */
@SuppressWarnings("all")
public class WordAnalyze {
    private String keyWord[] =
    {"break","include","begin","end","if","else","while","switch"};
    private char ch;
    //判断是否是关键字
    boolean isKey(String str)
    {
        for(int i = 0;i < keyWord.length;i++)
        {
            if(keyWord[i].equals(str))
                return true;
        }
        return false;
    }
    //判断是否是字母
    boolean isLetter(char letter)
    {
        if((letter >= 'a' && letter <= 'z') || (letter >= 'A' && letter <= 'Z'))
            return true;
        else
            return false;
    }
    //判断是否是数字
    boolean isDigit(char digit)
    {
        if(digit >= '0' && digit <= '9')
            return true;
        else
            return false;
    }
    //词法分析
    void analyze(char[] chars)
    {
        String arr = "";
        for(int i = 0;i< chars.length;i++) {
            ch = chars[i];
            arr = "";
            if(ch == ' ' || ch == '\t' || ch == '\n' || ch == '\r') {}
            else if(isLetter(ch)) {
                while(isLetter(ch) || isDigit(ch)) {
                    arr += ch;
                    ch = chars[++i];
                }
            }
        }
    }
}

```

```

}
//回退一个字符
i--;
if(isKey(arr)){
//关键字
System.out.println(arr+"\t4"+" \t 关键字");
}
else{
//标识符
System.out.println(arr+"\t4"+" \t 标识符");
}
}
else if(isDigit(ch) || (ch == '.'))
{
while(isDigit(ch) || (ch == '.' && isDigit(chars[++i])))
{
if(ch == '.') i--;
arr = arr + ch;
ch = chars[++i];
}
//属于无符号常数
System.out.println(arr+"\t5"+" \t 常数");
}
else switch(ch) {
//运算符
case '+':System.out.println(ch+"\t2"+" \t 运算符");break;
case '-':System.out.println(ch+"\t2"+" \t 运算符");break;
case '*':System.out.println(ch+"\t2"+" \t 运算符");break;
case '/':System.out.println(ch+"\t2"+" \t 运算符");break;
//分界符
case '(':System.out.println(ch+"\t3"+" \t 分界符");break;
case ')':System.out.println(ch+"\t3"+" \t 分界符");break;
case '[':System.out.println(ch+"\t3"+" \t 分界符");break;
case ']':System.out.println(ch+"\t3"+" \t 分界符");break;
case ';':System.out.println(ch+"\t3"+" \t 分界符");break;
case '{':System.out.println(ch+"\t3"+" \t 分界符");break;
case '}':System.out.println(ch+"\t3"+" \t 分界符");break;
//运算符
case '=': {
ch = chars[++i];
if(ch == '=')System.out.println("=="+"\t2"+" \t 运算符");
else {
System.out.println("=="+"\t2"+" \t 运算符");
i--;

```

```

    }
    }break;
    case ':' :{
    ch = chars[++i];
    if(ch == '=')System.out.println(":"+"\t2"+" \t 运算符");
    else {
    System.out.println(":"+"\t2"+" \t 运算符");
    i--;
    }
    }break;
    case '>' :{
    ch = chars[++i];
    if(ch == '=')System.out.println(">:"+"\t2"+" \t 运算符");
    else {
    System.out.println(">:"+"\t2"+" \t 运算符");
    i--;
    }
    }break;
    case '<' :{
    ch = chars[++i];
    if(ch == '=')System.out.println("<:"+"\t2"+" \t 运算符");
    else {
    System.out.println("<:"+"\t2"+" \t 运算符");
    i--;
    }
    }break;
    //无识别
    default: System.out.println(ch+"\t6"+" \t 无识别符");
    }
    }
    }

    public static void main(String[] args) throws Exception {
    File file = new File("D:\\code\\data.txt");//定义一个 file 对象，用来初始化
    FileReader
    FileReader reader = new FileReader(file);//定义一个 fileReader 对象，用来初始化
    BufferedReader
    int length = (int) file.length();
    //这里定义字符数组的时候需要多定义一个，因为词法分析器会遇到超前读取一个字符的时
    候，如果是最后一个
    //字符被读取，如果在读取下一个字符就会出现越界的异常
    char buf[] = new char[length+1];
    reader.read(buf);
    reader.close();
    new WordAnalyze().analyze(buf);

```

2、实验结果截图



```
35         if (digit >= '0' && digit <= '9')
36             return true;
```

Run: WordAnalyze ×

D:\environment\java\jdk-17.0.3.1\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2021.3.3\lib\idea_

int	4	标识符
+	2	运算符
123	5	常数
iyssfsf	4	标识符
+	2	运算符
-	2	运算符
)	3	分界符
(3	分界符
{	3	分界符
}	3	分界符
	6	无识别符

Process finished with exit code 0

3、实验总结

本次实验要求功能基本实现，可以完成题目的要求。通过本次实验我了解了如何设计，调试词法分析器，同时加深了对词法分析器原理的理解，了解了高级语言转化为目标代码的过程，加深了对编译原理的理解，了解了编译程序的实现编译程序的过程，巩固了课本上学到的理论知识。

	<p>该语法分析程序能判断输入的字符串是否符合上述文法，并能输出每一步骤的文法、已分析字符串、当前分析字符、未分析字符串。</p> <p>输入输出结果示例：</p> <pre>请输入算数表达式: i+i*i 步骤 文法 分析串 分析字符 剩余串 0 E -> TX i i i+i*i# 1 T -> FY i i i+i*i# 2 F -> i i i +i*i# 3 Y -> + i + +i*i# 4 X ->+TX i+ + i*i# 5 T -> FY i+ i i*i# 6 F -> i i+ i *i# 7 Y ->*FY i+i* * i# 8 F -> i i+i*i i # 9 Y -> ^ i+i*i # # 10 X -> ^ i+i*i # # 正确语句! 请输入算数表达式: i*/i 步骤 文法 分析串 分析字符 剩余串 0 E -> TX i i i*/i# 1 T -> FY i i i*/i# 2 F -> i i i */i# 3 Y ->*FY i* * /i# 分析失败!</pre>
实验时数	4 学时
实验步骤	(1) 定义模拟的简单语言的语法构成，调试语法分析程序； (2) 要求将用模拟语言书写的源程序进行语法分析； (3) 输出源程序运行结果，若分析有错误，须输出错误内容。

实验过程

1、实验代码

```
package com.clost.mybatis.pojo;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class Lexical {
    /*
    * 1 表示关键字
    * 2 表示标识符
    * 3 表示常数
    * 4 表示运算符
    * 5 表示界符
    * 6 表示字符串
    * */

    //关键字
    static String
```

```

[]keyWord={"private","protected","public","abstract","class","extends","final",
"implements",
"interface","native","new","static","strictfp","break","continue","return","do",
"while","if","else","for",
"instanceof","switch","case","default","boolean","byte","char","double","float",
"int","long","short",
"String","null","true","false","void","this","goto"};
//运算符
static String []operation={"+", "-", "*", "/", "%", "++", "--", "-=",
"*=","/=","&","|","^","~","<<",">>",">>>","==","!=",">","<","=",
">=","<=","&&","||","!","."};
//界符
static String []symbol={"",",",";",":", "(" ,")", "{", "}" };
static ArrayList<String> keyWords=null;
static ArrayList<String> operations=null;
static ArrayList<String> symbols=null;

//指向当前所读到字符串的位置的指针
static int p,lines;

public static void main(String []args) throws FileNotFoundException {
init();
File file=new File("D:\\code\\data.txt");
lines=1;
try(Scanner input=new Scanner(file)) {
while (input.hasNextLine()){
String str=input.nextLine();
analyze(str);
lines++;
}
}

}

//初始化把数组转换为 ArrayList
public static void init() {
keyWords=new ArrayList<>();
operations=new ArrayList<>();
symbols=new ArrayList<>();
Collections.addAll(keyWords, keyWord);
Collections.addAll(operations, operation);
Collections.addAll(symbols, symbol);
}

```

```

public static void analyze(String str){

    p=0;
    char ch;
    str=str.trim();
    for (;p<str.length();p++){
        ch=str.charAt(p);
        if (Character.isDigit(ch)){
            digitCheck(str);
        }else if (Character.isLetter(ch)||ch=='_'){
            letterCheck(str);
        }else if (ch=='"'){
            stringCheck(str);
        }
        else if (ch==' '){
            continue;
        }else {
            symbolCheck(str);
        }
    }
}

/*数字的识别
* 1、识别退出：
*   1.1、遇到空格符
*   1.2、遇到运算符或者界符
* 2、错误情况：
*   2.1、两个及以上小数点
*   2.2、掺杂字母
* */
public static void digitCheck(String str){
    String token= String.valueOf(str.charAt(p++));
    //判断数字的小数点是否有且是否大于1
    int flag=0;
    boolean err=false;
    char ch;
    for (;p<str.length();p++) {
        ch = str.charAt(p);
        if (ch==' '||(!Character.isLetterOrDigit(ch)&&ch!='.')) {
            break;
        }else if (err){
            token+=ch;
        }
    }
}

```



```

1&&(!Character.isLetterOrDigit(str.charAt(p))&&str.charAt(p)!='_')){
p--;
}
}

```

//符号的识别

```

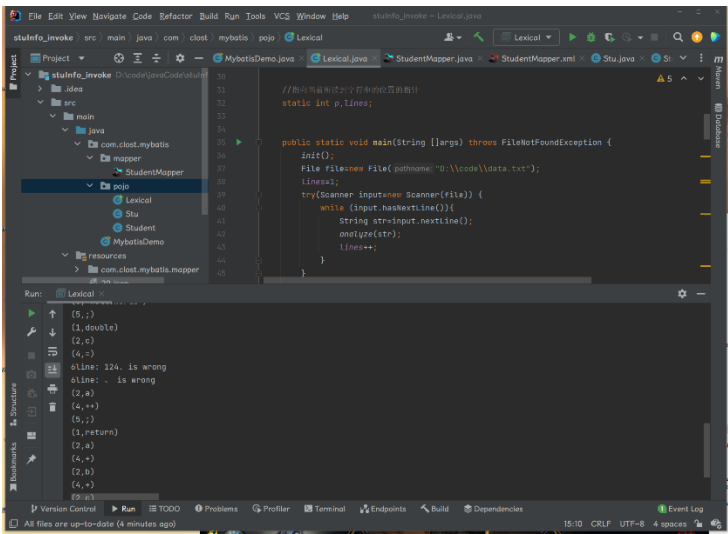
public static void symbolCheck(String str){
String token= String.valueOf(str.charAt(p++));
char ch;
if (symbols.contains(token)){
System.out.println("(" +5+" "+token+"");
p--;
}else {
if (operations.contains(token)){
if (p<str.length()){
ch=str.charAt(p);
if (operations.contains(token+ch)){
token+=ch;
p++;
if (p<str.length()){
ch=str.charAt(p);
if (operations.contains(token+ch)){
token+=ch;
System.out.println("(" +4+" "+token+"");
}else{
p--;
System.out.println("(" +4+" "+token+"");
}
}else{
System.out.println("(" +4+" "+token+"");
}
}
}else {
p--;
System.out.println("(" +4+" "+token+"");
}
}
}
}else {
p--;
System.out.println(lines+"line"+": "+token+" is wrong");
}
}
}
}

```

//字符串检查

```
public static void stringCheck(String str){
String token= String.valueOf(str.charAt(p++));
char ch;
for (;p<str.length();p++){
ch=str.charAt(p);
token+=ch;
if (ch==' '){
break;
}
}
if (token.charAt(token.length()-1)!=' '){
System.out.println(lines+"line"+": "+token+" is wrong");
}else {
System.out.println("(" +6+", "+token+"");
}
}
}
}
```

2、实验结果截图



3、实验总结

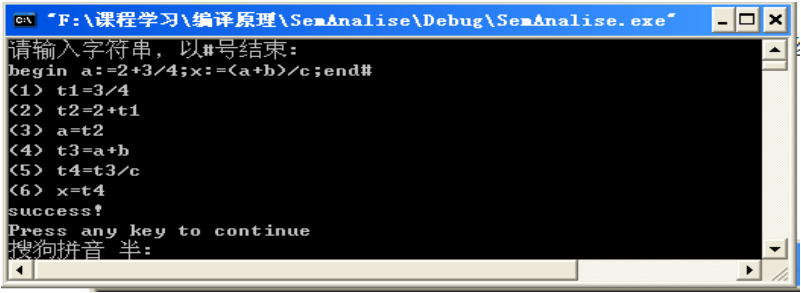
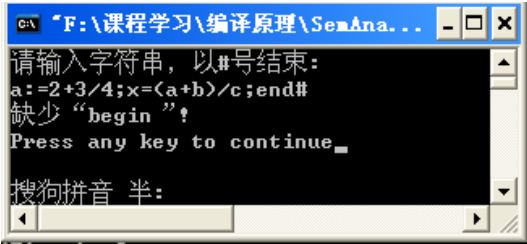
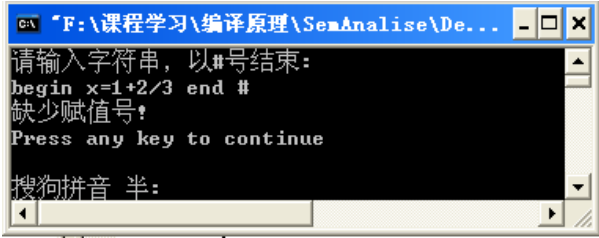
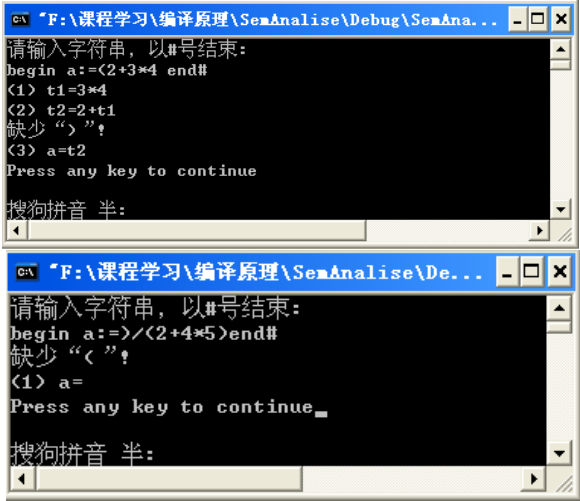
通过这次实验，我对语法分析器有了进一步的认识，把理论知识应用实践

同时也重新复习了 java 中的相关内容,加深了编程的知识的理解。通过语法分析实验，

我对高级语言的学习有了更深的认识，这对今后的学习有很大的帮助。

实验名称	实验三 语义分析器的设计
------	--------------

<p>目的要求</p>	<p>通过上机实验，加深对语法制导翻译的理解，掌握将语法分析所识别的语法成分转换为中间代码的语义翻译方法。</p>
<p>实验内容</p>	<p>对给定的程序通过语义分析器能够判断语句串是否正确。正确则输出三地址指令形式的四元式代码，错误则抛出错误信息。</p> <p>(1) 输入待分析的字符串。</p> <p>语法如下：</p> <p>a.关键字：begin,if,then,while,do,end.</p> <p>b.运算符和界符：:=+ - * / <=> >= <> = ; () #</p> <p>c.其他单词是标识符 (ID) 和整形常数 (NUM)：ID=letter(letter digit)*，NUM=digitdigit*</p> <p>d.空格由空白、制表符和换行符组成。空格一般用来分隔 ID、NUM、运算符、界符和关键字，词法分析阶段通常被忽略。</p> <p>(2) 扫描字符串，采用递归向下进行分析。</p> <p>主要函数如下：</p> <p>a.scanner()//词法分析函数，char token[8]用来存放构成单词符号的字符串；</p> <p>b.parser()//语法分析，在语法分析的基础上插入相应的语义动作：将输入串翻译成四元式序列。只对表达式、赋值语句进行翻译。</p> <p>c.emit(char *result,char *arg1,char *op,char *ag2)//该函数功能是生成一个三地址语句返回四元式表中。</p> <p>d.char *newtemp()//该函数返回一个新的临时变量名，临时变量名产生的顺序为 T1,T2,...。</p> <p>四元式表的结构如下：</p> <pre> struct {char result[8]; char ag1[8]; char op[8]; char ag2[8]; }quad[20]; </pre> <p>(3) 输出为三地址指令形式的四元式序列。</p> <p>例如：语句串 begin a:=2+3*4;x:=(a+b)/c;end#，</p> <p>输出的三地址指令如下：</p> <pre> t1=3*4 t2=2+t1 a=t2 t3=a+b t4=t3/c x=t4 </pre> <p>对于正确的语句串，例如：begin a:=2+3*4;x:=(a+b)/c; end#，运行结果如图：</p> <pre> 请输入字符串，以#号结束： begin a:=2+3*4;x:=(a+b)/c;end# <1> t1=3*4 <2> t2=2+t1 <3> a=t2 <4> t3=a+b <5> t4=t3/c <6> x=t4 success! </pre> <p>对于（”、“(”等均可做出错误判断并给出相应提示。Press any key to continue</p> <p>(4) 操作实例</p> <p>例如：对源程序 begin a:=2+3*4;x:=(a+b)/c; end#进行判断；</p>

	<p>首先运行程序，程序出现提示：“请输入字符串，以#号结束：”，在光标处输入 begin a:=2+3*4;x:=(a+b)/c;end#, 回车，结果如图：</p>  <p>程序给出了三地址码形式的四元式。</p> <p>1.2 错误处理</p> <p>(1) 如果用户在语句串开头处没有输入“begin”，程序提示“缺少 begin！”；</p>  <p>(2) 如果用户输入的语句串中缺少赋值符号（“:=”），程序提示“缺少赋值号！”；</p>  <p>(3) 如果用户输入的语句串中“(”和“)”不匹配，程序提示“缺少“)”或“缺少“(!””。</p> 
<p>实验时数</p>	<p>4 学时</p>
<p>实验步骤</p>	<p>(1) 给定任意语言实现以上内容。</p> <p>(2) 按给定示例进行测试。</p> <p>(3) 输出源程序运行结果，若分析有错误，须输出错误内容。</p>

实验过程

1、实验代码

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char prog[80], token[8];
char ch;
int syn, p, m, n, sum;
int kk=0, ii, N, nn=0;
int k=0, t, i=0;
char tt;
char *keywords[6]={"begin", "if", "then", "while", "do", "end"};
int scanner();
int parser();
int statement();
int sentence();
char *term();
char *factor();
char *expression();
void emit(char *result, char *ag1, char *op, char *ag2);
struct
{
char resulted[8];
char ag1ed[8];
char oped[8];
char ag2ed[8];
}quad[20];

int main()
{
p=0;
printf("请输入字符串，以#号结束:\n");
do
{
scanf("%c",&ch);
prog[p++]=ch;
}while(ch!='#');
p=0;
scanner();
parser();
}
```

```

char *newtemp(void)
{
char *P;
char M[8];
P=(char*)malloc(8);
k++;
itoa(k,M,10);
strcpy(P+1,M);
P[0]='t';
return(P);
}

```

```

int parser()
{
int schain=0;
kk=0;
if(syn==1)
{
scanner();
schain=sentence();
if(syn==6)
{
scanner();
if(syn==0&&(kk==0))
printf("success");
}
else
{
if(kk!=1)
printf("缺 end 错误");
kk=1;
}
}
else
{
printf("begin 错误");
kk=1;
}
return(schain);
}

```

```

int sentence()
{
int schain=0;

```

```

schain=statement();
while(syn==26)
{
    scanner();
    schain=statement();
}
return(schain);
}

```

```

int statement()
{
    char tt[8],eplace[8];
    int schain=0;
    switch(syn)
    {
        case 10:
            strcpy(tt,token);
            scanner();
            if(syn==18)
            {
                scanner();
                strcpy(eplace,expression());
                emit(tt,eplace,"", "");
                schain=0;
            }
            else
            {
                printf("缺少赋值号");
                kk=1;
            }
            return(schain);
            break;
    }
}

```

```

char *expression(void)
{
    char *tp,*ep2,*eplace,*tt;
    tp=(char*)malloc(12);
    ep2=(char*)malloc(12);
    eplace=(char*)malloc(12);
    tt=(char*)malloc(12);
    strcpy(eplace,term());
    while(syn==13||syn==14)

```

```

{
strcpy(tt, token);
scanner();
strcpy(ep2, term());
strcpy(tp, newtemp());
emit(tp, eplace, tt, ep2);
strcpy(eplace, tp);
}
return(eplace);
}

```

```

char *term(void)
{
char *tp, *ep2, *eplace, *tt;
tp=(char*)malloc(12);
ep2=(char*)malloc(12);
eplace=(char*)malloc(12);
tt=(char*)malloc(12);
strcpy(eplace, factor());
while(syn==15 || syn==16)
{
strcpy(tt, token);
scanner();
strcpy(ep2, factor());
strcpy(tp, newtemp());
emit(tp, eplace, tt, ep2);
strcpy(eplace, tp);
}
return(eplace);
}

```

```

char *factor(void)
{
char *fplace;
fplace=(char*)malloc(12);
strcpy(fplace, " ");
if(syn==10)
{
strcpy(fplace, token);
scanner();
}
else if(syn==11)
{
itoa(sum, fplace, 10);

```

```

scaner();
}
else if(syn==27)
{
scaner();
strcpy(fplace, expression());
if(syn==28)
scaner();
else
{
printf("' 错误");
kk=1;
}
}
else
{
printf("' 错误");
kk=1;
}
return(fplace);
}

```

```

void emit(char *result, char *ag1, char *op, char *ag2)
{
strcpy(quad[nn].resulted, result);
strcpy(quad[nn].ag1ed, ag1);
strcpy(quad[nn].oped, op);
strcpy(quad[nn].ag2ed, ag2);
printf("( %d\n",
    %s=%s%s%s\n",
nn+1, quad[nn].resulted, quad[nn].ag1ed, quad[nn].oped, quad[nn].ag2ed);
nn++;
}

```

```

scaner()
{
for(n=0;n<8;n++) token[n]=NULL;
ch=prog[p++]; m=0;
while(ch==' ') ch=prog[p++];
if((ch>='a')&&(ch<='z'))
{
while((ch>='a')&&(ch<='z') || (ch>='0')&&(ch<='9'))
{
token[m++]=ch;

```

```

ch=prog[p++];
}
token[m++]='\0';
p--;
syn=10;
for (n=0;n<6;n++)
if (strcmp(token, keywords[n])==0)
{
syn=n+1;
break;
}
}
else if (ch>='0' && ch<='9')
{
sum=0;
while ((ch>='0') && (ch<='9'))
{
sum=sum*10+(int)ch-'0';
ch=prog[p++];
}
p--;
syn=11;
}
else
switch(ch)
{
case '<':
m=0;
token[m++]=ch;
if (ch=='>')
{
syn=21;
token[m++]=ch;
} else if (ch=='=')
{
syn=22;
token[m++]=ch;
}
}
else
{
syn=20;
ch=prog[p++];
}
break;

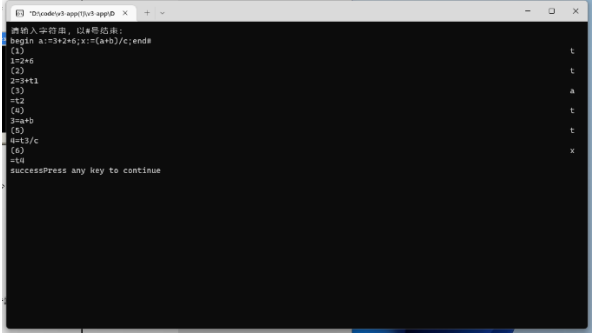
```

```

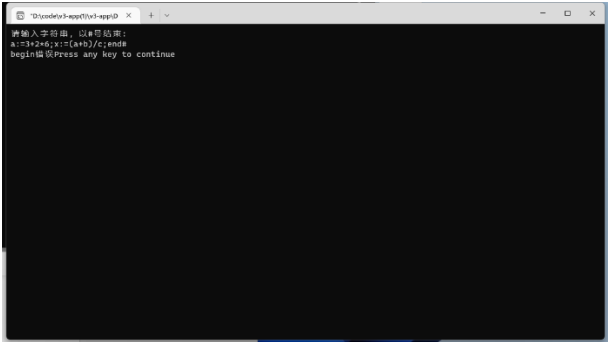
case'>':
m=0;
token[m++]=ch;
ch=prog[p++];
if(ch=='')
{
syn=24;
token[m++]=ch;
}
else
{
syn=23;
p--;
break;
}
break;
case':':
m=0;
token[m++]=ch;
ch=prog[p++];
if(ch=='')
{
syn=18;
token[m++]=ch;
}
else
{
syn=17;
p--;
}
break;
case'+':syn=13;token[0]=ch;break;
case'-':syn=14;token[0]=ch;break;
case'*':syn=15;token[0]=ch;break;
case'/':syn=16;token[0]=ch;break;
case '=':syn=25;token[0]=ch;break;
case';':syn=26;token[0]=ch;break;
case'(':syn=27;token[0]=ch;break;
case')':syn=28;token[0]=ch;break;
case'#':syn=0;token[0]=ch;break;
default:syn=-1;
}
return syn;
}

```

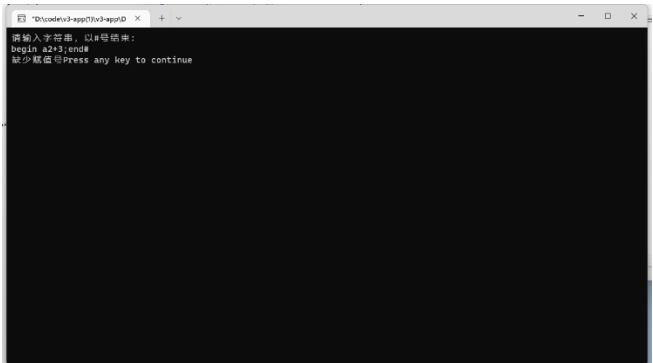

2、实验结果截图



```
请输入字符，以#号结束：
begin a:=5;x:=5;c:=a+b;c;ends
(1)
t:=a
(2)
c:=a+t1
(3)
t:=t2
(4)
t:=a+b
(5)
a:=t1/c
(6)
t:=t
successPress any key to continue
```



```
请输入字符，以#号结束：
a:=t1+x+b;c:=a+b/c;ends
beginPress any key to continue
```



```
请输入字符，以#号结束：
begin a:=5;ends
至少敲一个Press any key to continue
```

3、实验总结

通过此次实验，了解了语义分析的概念，对语法制导翻译有了新的认识，在编写和调试程序的过程中，对程序递归下降制导思想有了更深的理解，掌握了语法分析之别的语法成分变换为中间代码的方法。实验中也提高了自己的编程水平

实验名称	实验三 语法分析--LL（1）分析法
目的要求	加深对语法分析器工作过程的理解；加强对预测分析法实现语法分析程序的掌握；能够采用一种编程语言实现简单的语法分析程序；能够使用自己编写的分析程序对简单的程序段进行语法翻译。
实验内容	根据 LL(1) 语法分析算法的基本思想，设计一个对给定文法进行LL(1) 语法分析的程序，并用C、C++或 Java 语言编程实现。要求程序 能够对从键盘输入的任意字符串进行分析处理，判断出该输入串是否是给定文法的有效句子，并针对该串给出具体的 LL(1) 语法分析过程。
实验时数	4 学时
实验步骤	<ol style="list-style-type: none"> 1. 对语法规则有明确的定义； 2. 编写的分析程序能够对实验一的结果进行正确的语法分析； 3. 对于遇到的语法错误，能够做出简单的错误处理，给出简单的错误提示，保证顺利完成语法分析过程； 4. 实验报告要求用文法的形式对语法定义做出详细说明，说明语法分析程序的工作过程，说明错误处理的实现。

实验过程

1、实验代码

```

#include<stdlib.h>
#include<stdio.h>
#include<string.h>
/*****/

int count=0;          /*分解的产生式的个数*/
int number;           /*所有终结符和非终结符的总数*/
char start;           /*开始符号*/
char termin[50];      /*终结符号*/
char non_ter[50];     /*非终结符号*/
char v[50];           /*所有符号*/
char left[50];        /*左部*/
char right[50][50];   /*右部*/
char first[50][50], follow[50][50];    /*各产生式右部的 FIRST 和左部的 FOLLOW 集合*/
char first1[50][50];  /*所有单个符号的 FIRST 集合*/
char select[50][50];  /*各单个产生式的 SELECT 集合*/
char f[50], F[50];    /*记录各符号的 FIRST 和 FOLLOW 是否已求过*/
char empty[20];       /*记录可直接推出^的符号*/
char TEMP[50];        /*求 FOLLOW 时存放某一符号串的 FIRST 集合*/

```

```

int validity=1;          /*表示输入文法是否有效*/
int ll=1;                /*表示输入文法是否为 LL(1) 文法*/
int M[20][20];           /*分析表*/
char choose;             /*用户输入时使用*/
char empt[20];           /*求_emp() 时使用*/
char fo[20];             /*求 FOLLOW 集合时使用*/

```

```

/*****

```

判断一个字符是否在指定字符串中

```

*****/

```

```

int in(char c, char *p)
{
    int i;
    if(strlen(p)==0)
        return(0);
    for(i=0;;i++)
    {
        if(p[i]==c)
            return(1);      /*若在, 返回 1*/
        if(i==strlen(p))
            return(0);      /*若不在, 返回 0*/
    }
}

```

```

/*****

```

得到一个不是非终结符的符号

```

*****/

```

```

char c()
{
    char c='A';
    while(in(c, non_ter)==1)
        c++;
    return(c);
}

```

```

/*****

```

分解含有左递归的产生式

```

*****/

```

```

void recur(char *point)
{
    /*完整的产生式在 point[] 中*/
    int j, m=0, n=3, k;
    char temp[20], ch;
    ch=c();      /*得到一个非终结符*/
    k=strlen(non_ter);
    non_ter[k]=ch;
    non_ter[k+1]='\0';
}

```

```

for(j=0;j<=strlen(point)-1;j++)
{
if(point[n]==point[0])
{
/*如果 ‘|’ 后的首符号和左部相同*/
for(j=n+1;j<=strlen(point)-1;j++)
{
while(point[j]!='|'&&point[j]!='\0')
temp[m++]=point[j++];
left[count]=ch;
memcpy(right[count],temp,m);
right[count][m]=ch;
right[count][m+1]='\0';
m=0;
count++;
if(point[j]=='|')
{
n=j+1;
break;
}
}
}
else
{
/*如果 ‘|’ 后的首符号和左部不同*/
left[count]=ch;
right[count][0]='^';
right[count][1]='\0';
count++;
for(j=n;j<=strlen(point)-1;j++)
{
if(point[j]!='|')
temp[m++]=point[j];
else
{
left[count]=point[0];
memcpy(right[count],temp,m);
right[count][m]=ch;
right[count][m+1]='\0';
printf(" count=%d ",count);
m=0;
count++;
}
}
left[count]=point[0];
memcpy(right[count],temp,m);

```

```

right[count][m]=ch;
right[count][m+1]='\0' ;
count++;
m=0;
}
}
}

/*****
分解不含有左递归的产生式
*****/
void non_re(char *point)
{
int m=0, j;
char temp[20];
for(j=3; j<=strlen(point)-1; j++)
{
if(point[j]!='|')
temp[m++]=point[j];
else
{
left[count]=point[0];
memcpy(right[count], temp, m);
right[count][m]='\0' ;
m=0;
count++;
}
}
left[count]=point[0];
memcpy(right[count], temp, m);
right[count][m]='\0' ;
count++;
m=0;
}

/*****
读入一个文法
*****/
char grammer(char *t, char *n, char *left, char right[50][50])
{
char vn[50], vt[50];
char s;
char p[50][50];
int i, j, k;
printf("\n 请输入文法的非终结符号串：");
scanf("%s", vn);

```

```

getchar();
i=strlen(vn);
memcpy(n, vn, i);
n[i]='\0';
printf("请输入文法的终结符号串: ");
scanf("%s", vt);
getchar();
i=strlen(vt);
memcpy(t, vt, i);
t[i]='\0';
printf("请输入文法的开始符号: ");
scanf("%c", &s);
getchar();
printf("请输入文法产生式的条数: ");
scanf("%d", &i);
getchar();
for(j=1; j<=i; j++)
{
printf("请输入文法的第%d 条 (共%d 条) 产生式: ", j, i);
scanf("%s", p[j-1]);
getchar();
}
for(j=0; j<=i-1; j++)
if(p[j][1]!='-' || p[j][2]!='>')
{ printf("\ninput error!");
validity=0;
return('\0');
} /*检测输入错误*/
for(k=0; k<=i-1; k++)
{ /*分解输入的各产生式*/
if(p[k][3]==p[k][0])
recur(p[k]);
else
non_re(p[k]);
}
return(s);
}

/*****
将单个符号或符号串并入另一符号串
*****/

void merge(char *d, char *s, int type)
{ /*d 是目标符号串, s 是源串, type=1, 源串中的 ‘ ^ ’ 一并并入目串;
type=2, 源串中的 ‘ ^ ’ 不并入目串*/
int i, j;

```

```

for(i=0;i<=strlen(s)-1;i++)
{
if(type==2&&s[i]=='^')
;
else
{
for(j=0;;j++)
{
if(j<strlen(d)&&s[i]==d[j])
break;
if(j==strlen(d))
{
d[j]=s[i];
d[j+1]='\0';
break;
}
}
}
}
}

/*****
求所有能直接推出^的符号
*****/

void emp(char c)
{
/*即求所有由 ‘ ^ ’ 推出的符号*/
char temp[10];
int i;
for(i=0;i<=count-1;i++)
{
if(right[i][0]==c&&strlen(right[i])==1)
{
temp[0]=left[i];
temp[1]='\0';
merge(empty, temp, 1);
emp(left[i]);
}
}
}

/*****
求某一符号能否推出 ‘ ^ ’
*****/

int _emp(char c)
{
/*若能推出，返回 1；否则，返回 0*/
int i, j, k, result=1, mark=0;

```

```

char temp[20];
temp[0]=c;
temp[1]='\0';
merge(empty, temp, 1);
if(in(c, empty)==1)
return(1);
for(i=0;;i++)
{
if(i==count)
return(0);
if(left[i]==c)          /*找一个左部为 c 的产生式*/
{
j=strlen(right[i]);    /*j 为右部的长度*/
if(j==1&&in(right[i][0], empty)==1)
return(1);
else if(j==1&&in(right[i][0], termin)==1)
return(0);
else
{
for(k=0;k<=j-1;k++)
if(in(right[i][k], empty)==1)
mark=1;
if(mark==1)
continue;
else
{
for(k=0;k<=j-1;k++)
{
result*=_emp(right[i][k]);
temp[0]=right[i][k];
temp[1]='\0';
merge(empty, temp, 1);
}
}
}
if(result==0&&i<count)
continue;
else if(result==1&&i<count)
return(1);
}
}
}

/*****
判断读入的文法是否正确

```



```

*****/
int judge()
{
    int i, j;
    for(i=0; i<=count-1; i++)
    {
        if(in(left[i], non_ter)==0)
        {
            /*若左部不在非终结符中，报错*/
            printf("\nerror1!");
            validity=0;
            return(0);
        }
        for(j=0; j<=strlen(right[i])-1; j++)
        {
            if(in(right[i][j], non_ter)==0&&in(right[i][j], termin)==0&&right[i][j]!='^')
            {
                /*若右部某一符号不在非终结符、终结符中且不为 '^'，报错*/
                printf("\nerror2!");
                validity=0;
                return(0);
            }
        }
    }
    return(1);
}

/*****
求单个符号的 FIRST
*****/
void first2(int i)
{
    /*i 为符号在所有输入符号中的序号*/
    char c, temp[20];
    int j, k, m;
    c=v[i];
    char ch='^';
    emp(ch);
    if(in(c, termin)==1) /*若为终结符*/
    {
        first1[i][0]=c;
        first1[i][1]='\0';
    }
    else if(in(c, non_ter)==1) /*若为非终结符*/
    {
        for(j=0; j<=count-1; j++)
        {
            if(left[j]==c)

```

```

{
if (in(right[j][0], termin)==1 || right[j][0]=='^')
{
temp[0]=right[j][0];
temp[1]='\0';
merge(first1[i], temp, 1);
}
else if (in(right[j][0], non_ter)==1)
{
if (right[j][0]==c)
continue;
for (k=0; ;k++)
if (v[k]==right[j][0])
break;
if (f[k]=='0')
{
first2(k);
f[k]='1';
}
merge(first1[i], first1[k], 2);
for (k=0; k<=strlen(right[j])-1; k++)
{
empt[0]='\0';
if (_emp(right[j][k])==1 && k<strlen(right[j])-1)
{
for (m=0; ;m++)
if (v[m]==right[j][k+1])
break;
if (f[m]=='0')
{
first2(m);
f[m]='1';
}
merge(first1[i], first1[m], 2);
}
else if (_emp(right[j][k])==1 && k==strlen(right[j])-1)
{
temp[0]='^';
temp[1]='\0';
merge(first1[i], temp, 1);
}
else
break;
}
}

```

```

}
}
}
}
f[i]='1';
}
/*****
求各产生式右部的 FIRST
*****/
void FIRST(int i, char *p)
{
    int length;
    int j, k, m;
    char temp[20];
    length=strlen(p);
    if(length==1) /*如果右部为单个符号*/
    {
        if(p[0]!='^')
        {
            if(i>=0)
            {
                first[i][0]='^';
                first[i][1]='\0';
            }
        }
        else
        {
            TEMP[0]='^';
            TEMP[1]='\0';
        }
    }
    else
    {
        for(j=0;;j++)
            if(v[j]==p[0])
                break;
        if(i>=0)
        {
            memcpy(first[i], first1[j], strlen(first1[j]));
            first[i][strlen(first1[j])]='\0';
        }
        else
        {
            memcpy(TEMP, first1[j], strlen(first1[j]));
            TEMP[strlen(first1[j])]='\0';

```

```

}
}
}
else /*如果右部为符号串*/
{
for(j=0;;j++)
if(v[j]==p[0])
break;
if(i>=0)
merge(first[i],first1[j],2);
else
merge(TEMP,first1[j],2);
for(k=0;k<=length-1;k++)
{
empt[0]='\0';
if(_emp(p[k])==1&&k<length-1)
{
for(m=0;;m++)
if(v[m]==right[i][k+1])
break;
if(i>=0)
merge(first[i],first1[m],2);
else
merge(TEMP,first1[m],2);
}
else if(_emp(p[k])==1&&k==length-1)
{

temp[0]='^';
temp[1]='\0';
if(i>=0)
merge(first[i],temp,1);
else
merge(TEMP,temp,1);
}
else if(_emp(p[k])==0)
break;
}
}
}
/*****
求各产生式左部的 FOLLOW
*****/
void FOLLOW(int i)

```

```

{
int j, k, m, n, result=1;
char c, temp[20];
c=non_ter[i];          /*c 为待求的非终结符*/
temp[0]=c;
temp[1]='\0';
merge(fo, temp, 1);
if(c==start)
{
/*若为开始符号*/
temp[0]='#';
temp[1]='\0';
merge(follow[i], temp, 1);
}
for(j=0; j<=count-1; j++)
{
if(in(c, right[j])==1)    /*找一个右部含有 c 的产生式*/
{
for(k=0; ; k++)
if(right[j][k]==c)
break;          /*k 为 c 在该产生式右部的序号*/
for(m=0; ; m++)
if(v[m]==left[j])
break;          /*m 为产生式左部非终结符在所有符号中的序号*/
if(k==strlen(right[j])-1)
{
/*如果 c 在产生式右部的最后*/
if(in(v[m], fo)==1)
{
merge(follow[i], follow[m], 1);
continue;
}
if(F[m]=='0')
{
FOLLOW(m);
F[m]='1';
}
merge(follow[i], follow[m], 1);
}
else
{
/*如果 c 不在产生式右部的最后*/
for(n=k+1; n<=strlen(right[j])-1; n++)
{
empt[0]='\0';
result*=_emp(right[j][n]);
}
}
}
}
}

```

```

if(result==1)
{
    /*如果右部 c 后面的符号串能推出^*/
    if(in(v[m],fo)==1)
    {
        /*避免循环递归*/
        merge(follow[i], follow[m], 1);
        continue;
    }
    if(F[m]=='0')
    {
        FOLLOW(m);
        F[m]='1';
    }
    merge(follow[i], follow[m], 1);
}
for(n=k+1;n<=strlen(right[j])-1;n++)
temp[n-k-1]=right[j][n];
temp[strlen(right[j])-k-1]='\0';
FIRST(-1, temp);
merge(follow[i], TEMP, 2);
}
}
}
F[i]='1';
}

/*****
判断读入文法是否为一个 LL(1) 文法
*****/
int ll1()
{
    int i, j, length, result=1;
    char temp[50];
    for(j=0; j<=49; j++)
    {
        /*初始化*/
        first[j][0]='\0';
        follow[j][0]='\0';
        first1[j][0]='\0';
        select[j][0]='\0';
        TEMP[j]='\0';
        temp[j]='\0';
        f[j]='0';
        F[j]='0';
    }
    for(j=0; j<=strlen(v)-1; j++)

```

```

first2(j);                /*求单个符号的 FIRST 集合*/
printf("\nfirst1:");
for(j=0;j<=strlen(v)-1;j++)
printf("%c:%s  ",v[j],first1[j]);
printf("\nempty:%s",empty);
printf("\n:::\n_emp:");
for(j=0;j<=strlen(v)-1;j++)
printf("%d  ",_emp(v[j]));
for(i=0;i<=count-1;i++)
FIRST(i,right[i]);        /*求 FIRST*/
printf("\n");
for(j=0;j<=strlen(non_ter)-1;j++)
{
    /*求 FOLLOW*/
    if(fo[j]==0)
    {
        fo[0]='\0';
        FOLLOW(j);
    }
}
printf("\nfirst:");
for(i=0;i<=count-1;i++)
printf("%s  ",first[i]);
printf("\nfollow:");
for(i=0;i<=strlen(non_ter)-1;i++)
printf("%s  ",follow[i]);
for(i=0;i<=count-1;i++)
{
    /*求每一产生式的 SELECT 集合*/
    memcpy(select[i],first[i],strlen(first[i]));
    select[i][strlen(first[i])]='\0';
    for(j=0;j<=strlen(right[i])-1;j++)
    result*=_emp(right[i][j]);
    if(strlen(right[i])==1&&right[i][0]=='^')
    result=1;
    if(result==1)
    {
        for(j=0;;j++)
        if(v[j]==left[i])
        break;
        merge(select[i],follow[j],1);
    }
}
printf("\nselect:");
for(i=0;i<=count-1;i++)
printf("%s  ",select[i]);

```

```

memcpy(temp, select[0], strlen(select[0]));
temp[strlen(select[0])] = '\0';
for (i=1; i<=count-1; i++)
{
    /*判断输入文法是否为 LL(1) 文法*/
    length = strlen(temp);
    if (left[i] == left[i-1])
    {
        merge(temp, select[i], 1);
        if (strlen(temp) < length + strlen(select[i]))
            return(0);
    }
    else
    {
        temp[0] = '\0';
        memcpy(temp, select[i], strlen(select[i]));
        temp[strlen(select[i])] = '\0';
    }
}
return(1);
}

```

```

/*****
构造分析表 M
*****/

```

```

void MM()
{
    int i, j, k, m;
    for (i=0; i<=19; i++)
        for (j=0; j<=19; j++)
            M[i][j] = -1;
    i = strlen(termin);
    termin[i] = '#'; /*将#加入终结符数组*/
    termin[i+1] = '\0';
    for (i=0; i<=count-1; i++)
    {
        for (m=0; m++)
            if (non_ter[m] == left[i])
                break; /*m 为产生式左部非终结符的序号*/
        for (j=0; j<=strlen(select[i])-1; j++)
        {
            if (in(select[i][j], termin) == 1)
            {
                for (k=0; k++)
                    if (termin[k] == select[i][j])

```



```

break;          /*k 为产生式右部终结符的序号*/
M[m][k]=i;
}
}
}
}

/*****
总控算法
*****/
void syntax()
{
int i, j, k, m, n, p, q;
char ch;
char S[50], str[50];
printf("请输入该文法的句型: ");
scanf("%s", str);
getchar();
i=strlen(str);
str[i]='#';
str[i+1]='\0';
S[0]='#';
S[1]=start;
S[2]='\0';
j=0;
ch=str[j];
while(1)
{
if(in(S[strlen(S)-1], termin)==1)
{
if(S[strlen(S)-1]!=ch)
{
printf("\n 该符号串不是文法的句型!");
return;
}
else if(S[strlen(S)-1]=='#')
{
printf("\n 该符号串是文法的句型.");
return;
}
else
{
S[strlen(S)-1]='\0';
j++;

```

```

ch=str[j];
}
}
else
{
for(i=0;;i++)
if(non_ter[i]==S[strlen(S)-1])
break;
for(k=0;;k++)
{
if(termin[k]==ch)
break;
if(k==strlen(termin))
{
printf("\n 词法错误! ");
return;
}
}
if(M[i][k]==-1)
{
printf("\n 语法错误! ");
return;
}
else
{
m=M[i][k];
if(right[m][0]=='^')
S[strlen(S)-1]='\0';
else
{
p=strlen(S)-1;
q=p;
for(n=strlen(right[m])-1;n>=0;n--)
S[p++]=right[m][n];
S[q+strlen(right[m])]='\0';
}
}
}
printf("\nS:%s str:",S);
for(p=j;p<=strlen(str)-1;p++)
printf("%c",str[p]);
printf(" ");
}
}

```

```
/******
```

```
一个用户调用函数
```

```
*****
```

```
void menu()
```

```
{
```

```
    syntax();
```

```
    printf("\n 是否继续? (y or n):");
```

```
    scanf("%c",&choose);
```

```
    getchar();
```

```
    while(choose=='y')
```

```
    {
```

```
        menu();
```

```
    }
```

```
}
```

```
/******
```

```
主函数
```

```
*****
```

```
int main()
```

```
{
```

```
    int i, j;
```

```
    start=grammer(termin, non_ter, left, right);
```

```
/*读入一个文法*/
```

```
    printf("count=%d", count);
```

```
    printf("\nstart:%c", start);
```

```
    strcpy(v, non_ter);
```

```
    strcat(v, termin);
```

```
    printf("\nv:%s", v);
```

```
    printf("\nnon_ter:%s", non_ter);
```

```
    printf("\ntermin:%s", termin);
```

```
    printf("\nright:");
```

```
    for(i=0;i<=count-1;i++)
```

```
        printf("%s", right[i]);
```

```
    printf("\nleft:");
```

```
    for(i=0;i<=count-1;i++)
```

```
        printf("%c", left[i]);
```

```
    if(validity==1)
```

```
        validity=judge();
```

```
    printf("\nvalidity=%d", validity);
```

```
    if(validity==1)
```

```
    {
```

```
        printf("\n 文法有效");
```

```
    }
}
```

2、实验结果截图

[illegible]

```

1: #M01 str:=i;end
2: #M02 str:=i+1;end
3: #M03 str:=i+10;end
4: #M04 str:=i+100;end
5: #M05 str:=i+1000;end
6: #M06 str:=i+10000;end
7: #M07 str:=i;end
8: #M08 str:=i;end
9: #M09 str:=i;end
10: #M10 str:=i;end
11: #M11 str:=i;end
12: #M12 str:=i;end
13: #M13 str:=i;end
14: #M14 str:=i;end
15: #M15 str:=i;end
16: #M16 str:=i;end
17: #M17 str:=i;end
18: #M18 str:=i;end
19: #M19 str:=i;end
20: #M20 str:=i;end
21: #M21 str:=i;end
22: #M22 str:=i;end
23: #M23 str:=i;end
24: #M24 str:=i;end
25: #M25 str:=i;end
26: #M26 str:=i;end
27: #M27 str:=i;end
28: #M28 str:=i;end
29: #M29 str:=i;end
30: #M30 str:=i;end
31: #M31 str:=i;end
32: #M32 str:=i;end
33: #M33 str:=i;end
34: #M34 str:=i;end
35: #M35 str:=i;end
36: #M36 str:=i;end
37: #M37 str:=i;end
38: #M38 str:=i;end
39: #M39 str:=i;end
40: #M40 str:=i;end
41: #M41 str:=i;end
42: #M42 str:=i;end
43: #M43 str:=i;end
44: #M44 str:=i;end
45: #M45 str:=i;end
46: #M46 str:=i;end
47: #M47 str:=i;end
48: #M48 str:=i;end
49: #M49 str:=i;end
50: #M50 str:=i;end
51: #M51 str:=i;end
52: #M52 str:=i;end
53: #M53 str:=i;end
54: #M54 str:=i;end
55: #M55 str:=i;end
56: #M56 str:=i;end
57: #M57 str:=i;end
58: #M58 str:=i;end
59: #M59 str:=i;end
60: #M60 str:=i;end
61: #M61 str:=i;end
62: #M62 str:=i;end
63: #M63 str:=i;end
64: #M64 str:=i;end
65: #M65 str:=i;end
66: #M66 str:=i;end
67: #M67 str:=i;end
68: #M68 str:=i;end
69: #M69 str:=i;end
70: #M70 str:=i;end
71: #M71 str:=i;end
72: #M72 str:=i;end
73: #M73 str:=i;end
74: #M74 str:=i;end
75: #M75 str:=i;end
76: #M76 str:=i;end
77: #M77 str:=i;end
78: #M78 str:=i;end
79: #M79 str:=i;end
80: #M80 str:=i;end
81: #M81 str:=i;end
82: #M82 str:=i;end
83: #M83 str:=i;end
84: #M84 str:=i;end
85: #M85 str:=i;end
86: #M86 str:=i;end
87: #M87 str:=i;end
88: #M88 str:=i;end
89: #M89 str:=i;end
90: #M90 str:=i;end
91: #M91 str:=i;end
92: #M92 str:=i;end
93: #M93 str:=i;end
94: #M94 str:=i;end
95: #M95 str:=i;end
96: #M96 str:=i;end
97: #M97 str:=i;end
98: #M98 str:=i;end
99: #M99 str:=i;end
100: #M100 str:=i;end

```

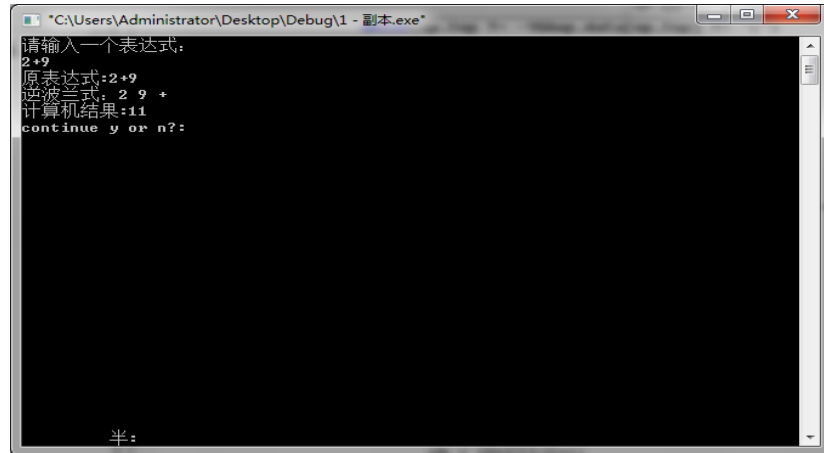
3、实验总结

通过本次实验，在程序分析的过程中，了解了 LL(1) 算法的基本思想。通过自己动手实现这一功能，用实践加强了对理论的认知，从课本中对理论的认知有一定限度，亲自动手实验的时候，发现去实现并不容易。实现过程中，通过查找代码，渐渐对文法有了更加深刻的认识，实践中能学到课本中学不到的东西。

实验名称	实验四 综合实验
目的要求	<p>1、掌握有穷状态自动机的概念；</p> <p>2、掌握有穷状态自动机的存储及表示方法；</p> <p>3、掌握有穷状态自动机与正则文法之间的联系。</p> <p>4、对单词的构词规则有明确的定义；</p> <p>5、借助逆波兰表达式实现计算器。</p>
实验内容	<p>综合实验一</p> <p>1、识别有穷状态自动机是确定的还是非确定的；</p> <p>2、构造的有穷状态自动机以相应的五元组形式输出。</p> <p>以下示例为一种输出，可以有其他形式：</p> <div><pre>请输入规则个数：3 请输入文法： Z::=Za!Aa!Bb A::=Ba!Za!a B::=Ab!Ba!b 是非确定的有穷状态自动机，即NFA 构造的有穷状态自动机为： NFA N = (K, Σ, M, <S>, <Z>) 其中， K = {S, Z, A, B} Σ = {a, b} M: M(S,a)={A} M(S,b)={B} M(Z,a)={Z A} M(Z,b)={} M(A,a)={Z} M(A,b)={B} M(B,a)={A B} M(B,b)={Z} 请按任意键继续...</pre></div> <div><pre>请输入规则个数：3 请输入文法： Z::=Za!Aa!Bb A::=Ba!a B::=Ab!b 是确定的有穷状态自动机，即DFA 构造的有穷状态自动机为： DFA D = (K, Σ, M, <S>, <Z>) 其中， K = {S, Z, A, B} Σ = {a, b} M: M(S,a)={A} M(S,b)={B} M(Z,a)={Z} M(Z,b)={} M(A,a)={Z} M(A,b)={B} M(B,a)={A} M(B,b)={Z} 请按任意键继续...</pre></div> <p>综合实验二</p> <p>设计内容：</p> <p>计算器的功能要求如下：可以支持加(+)、减(-)、乘(*)、除(/)运算,如 3+4-5*2/2;支持括号运算，如 (4+5)*5/8。用户输入表达式后，转化为逆波兰式并执行计算，最后输出该表达式的结果。</p>

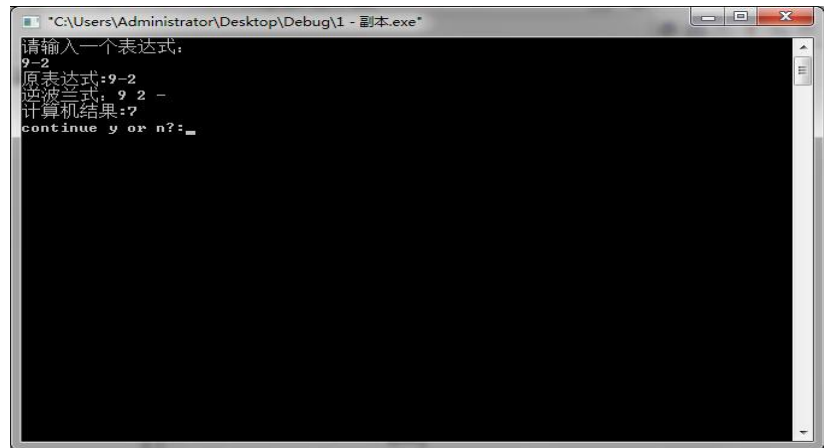
设计要求:

- 1、将非后缀式用来表示的算术表达式转换为用逆波兰式来表示的算术表达式，并计算用逆波兰式来表示的算术表达式的值。
- 2、掌握利用算符优先分析法完成中缀表达式到逆波兰式的转化。



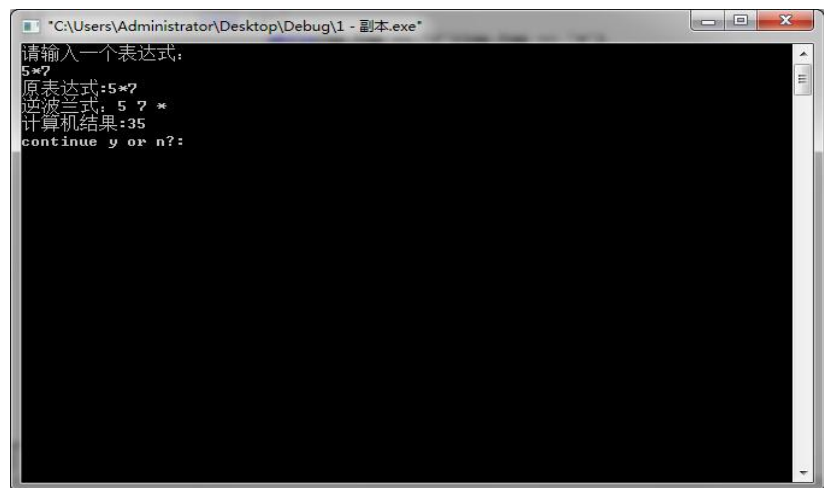
```
请输入一个表达式:
2+9
原表达式:2+9
逆波兰式: 2 9 +
计算机结果:11
continue y or n?:
半:
```

加法



```
请输入一个表达式:
9-2
原表达式:9-2
逆波兰式: 9 2 -
计算机结果:7
continue y or n?:
```

减法



```
请输入一个表达式:
5*7
原表达式:5*7
逆波兰式: 5 7 *
计算机结果:35
continue y or n?:
```

乘法

	<div></div> <p>除法</p> <div></div> <p>混合运算</p>
实验时数	4 学时
实验步骤	(1) 给定任意语言实现以上内容。 (2) 按给定示例进行测试。 (3) 输出源程序运行结果，若分析有错误，须输出错误内容。

实验过程

1、实验代码

综合实验一

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
int main()
{
char p[30][30];
```



```

char q[30][30];
int line=0;
int n;
int i,j;
int count=0;
int k,t=0;
int flag=0;
int l,m=0;
char VN[30]={'\0'};
char VT[30]={'\0'};
printf("请输入规则数:");
scanf("%d",&n);
line=n;
for(i=0;i<30;i++)
for(j=0;j<30;j++)
{
p[i][j]='\0';
q[i][j]='\0';
}
printf("请输入文法:\n");
for(i=0;i<line;i++)
{
scanf("%s",p[i]);
}

l=0;
m=0;
for(i=0;i<line;i++)
{
for(j=0;j<30&&(p[i][j]!='\0');j++)
{

if(p[i][j]<='z' && p[i][j]>='a' || (p[i][j]<='9' && p[i][j]>='0'))
{
flag=0;
for(t=0;VN[t]!='\0';t++)
{
if(VN[t]==p[i][j])
{
flag=1;break;
}
}
}
if(flag==0)
{

```

```

VN[l]=p[i][j];
l++;
}
}

if (p[i][j]<=' Z' && p[i][j]>=' A' )
{
flag=0;
for (t=0;t<30&&(VT[t]!=' \0' );t++)
{
if (VT[t]==p[i][j])
{
flag=1;
break;
}
}
if (flag==0)
{
VT[m]=p[i][j];
m++;
}
}
}
}

count=0;
k=0;
for (i=0;i<line;i++)
{
for (j=4;j<30&&(p[i][j]!=' \0' );j++)
{
if ((p[i][j]<=' z' && p[i][j]>=' a' ) || (p[i][j]<=' Z' && p[i][j]>=' A' ) || (p[i][j]<=' 9' && p[i][j]>=' 0' ))
{
q[count][k]=p[i][j];
k++;
}
else
{
count++;
k=0;
}
}
count++;
}

```

```

k=0;
}
flag=0;
for(i=0;i<count;i++)
{
for(j=i+1;j<count;j++)
{
if(strcmp(q[i],q[j])==0)
{
flag=1;
break;
}
}
}
if(flag==1)
{
printf("是非确定的有穷状态自动机, 即 NFA\n\n");
printf("构造的有穷状态自动机为:\n");
printf("NFA N=(K,  $\Sigma$ , M, {S}, {Z})\n");
}
else
{
printf("是确定的有穷状态自动机, 即 DFA\n\n\n");
printf("构造的有穷状态自动机为:\n");
printf("DFA N=(K,  $\Sigma$ , M, {S}, {Z})\n");
}
printf("其中, \nK={S}");
for(i=0;i<30&&(VT!='\0');i++)
{
printf(",%c",VT[i]);
}
printf("}\n");
printf("E={");
for(i=0;i<30&&(VN[i]!='\0');i++)
{
printf("%c ",VN[i]);
}
printf("}\n");

k=0;
count=0;
for(i=0;i<line;i++)
{
j=4;

```

```

while(p[i][j]!='\0')
{
if(k<4)
{
q[count][k]=p[i][k];
k++;
}
else
{
if((p[i][j]<='z' && p[i][j]>='a') || (p[i][j]<='Z' && p[i][j]>='A') || (p[i][j]<='9' && p[i][j]>='0'))
{
q[count][k]=p[i][j];
k++;
j++;
}
if(p[i][j]=='l')
{
count++;
k=0;
j++;
}
}
count++;
k=0;
}
printf("\n");

printf("M:\n");
l=0;
while(VN[l]!='\0')
{
printf("M(S,%c)={",VN[l]);
for(i=0;i<30;i++)
{
for(j=4;j<30&&(q[i][j]!='\0');j++)
{
if(VN[l]==q[i][j]&&(q[i][j+1]!='\0')&&(q[i][j-1]==''))
printf("%c",q[i][0]);
}
}
printf("}\t");
l++;
}

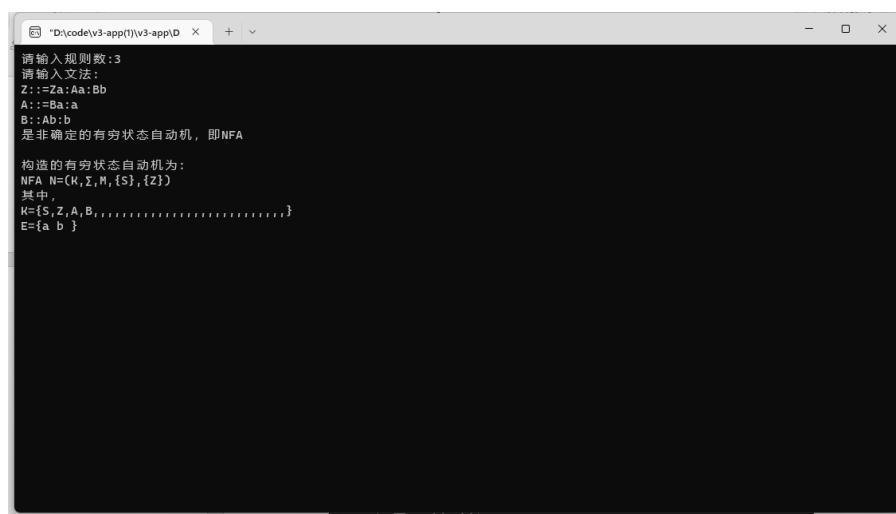
```

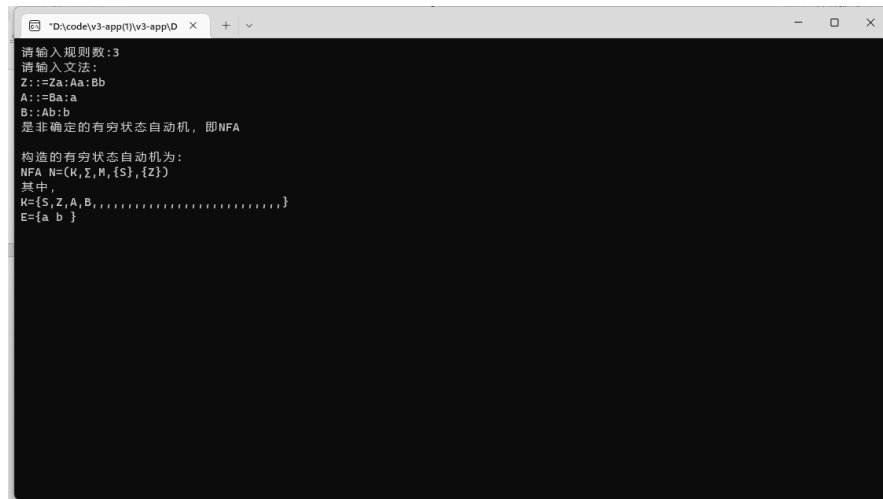
```

}
printf("\n");
l=0;k=0;
while(VT[k]!='\0')
{
l=0;
while(VN[l]!='\0')
{
printf("M(%c,%c)={",VT[k],VN[l]);
for(i=0;i<30;i++)
{
for(j=4;j<30&&(q[i][j]!='\0');j++)
{
if(VT[k]==q[i][j]&&VN[l]==q[i][j+1])
printf("%c",q[i][0]);
}
}
printf("}\t");
l++;
}
k++;
printf("\n");
}
system("pause");
}

```

2、实验结果截图





综合实验二

```
#include<stdio.h>
#include<stdlib.h>
#define MaxSize 99

void translate(char str[],char exp[])
{
    struct
    {
        char data[MaxSize];
        int top;
    }op;
    char ch;
    int i = 0,t = 0;
    op.top = -1;
    ch = str[i];
    i++;
    while(ch != '\0')
    {
        switch(ch)
        {
            case '(' :
                op.top++;op.data[op.top]=ch;
                break;
            case ')' :
                while(op.data[op.top] != '(')
                {
                    exp[t]=op.data[op.top];
                    op.top--;
                    t++;
                }
            }
        }
    }
}
```

```

op.top--;
break;
case '+':
case '-':
while(op.top != -1 && op.data[op.top] != '(')
{
exp[t] = op.data[op.top];
op.top--;
t++;
}
op.top++;
op.data[op.top] = ch;
break;
case '*':
case '/':
while(op.data[op.top] == '/' || op.data[op.top] == '*')
{
exp[t] = op.data[op.top];
op.top--;
t++;
}
op.top++;
op.data[op.top] = ch;
break;
case ' ':
break;
default:
while(ch >= '0' && ch <= '9')
{
exp[t] = ch; t++;
ch = str[i]; i++;
}
i--;
exp[t] = ' ';
t++;
}
ch = str[i];
i++;
}
while(op.top != -1)
{
exp[t] = op.data[op.top];
t++;
op.top--;

```

```

}
exp[t] = '\0';
}

```

```

float cal_value(char exp[])
{
    struct
    {
        float data[MaxSize];
        int top;
    }st;
    float d;
    char ch;
    int t = 0;
    st.top = -1;
    ch = exp[t];
    t++;
    while(ch != '\0')
    {
        switch(ch)
        {
            case '+':
                st.data[st.top-1] = st.data[st.top-1]+st.data[st.top];
                st.top--;
                break;
            case '-':
                st.data[st.top-1] = st.data[st.top-1]-st.data[st.top];
                st.top--;
                break;
            case '*':
                st.data[st.top-1] = st.data[st.top-1]*st.data[st.top];
                st.top--;
                break;
            case '/':
                if(st.data[st.top] != 0)
                    st.data[st.top-1]=st.data[st.top-1]/st.data[st.top];
                else
                {
                    printf("\n\terror");
                }
                st.top--;
                break;
            default:

```



```

d=0;
while(ch >= '0' && ch <= '9')
{
d = 10*d+ch-'0';
ch = exp[t];
t++;
}
st.top++;
st.data[st.top] = d;
}
ch = exp[t];
t++;
}
return st.data[st.top];
}

int main()
{
char ch;
while(1)
{
char str[MaxSize], exp[MaxSize];
printf("请输入一个表达式: \n");
gets(str);
printf("原表达式:%s\n", str);
translate(str, exp);
printf("逆波兰式: %s\n", exp);
printf("计算机结果:%g\n", cal_value(exp));
printf("continue y or n?:");
scanf("%c", &ch);
if(ch=='Y' || ch=='y')
{
gets(str);
}
else
{
break;
}
}
//system("pause");
return 0;
}

```

2、实验结果截图

```
"D:\code\v3-app(1)\v3-app\D  x + v
请输入一个表达式:
1+1
原表达式:1+1
逆波兰式: 1 1 +
计算机结果:2
continue y or n?:y
请输入一个表达式:
2-1
原表达式:2-1
逆波兰式: 2 1 -
计算机结果:1
continue y or n?:y
请输入一个表达式:
2*8
原表达式:2*8
逆波兰式: 2 8 *
计算机结果:16
continue y or n?:y
请输入一个表达式:
8/2
原表达式:8/2
逆波兰式: 8 2 /
计算机结果:4
continue y or n?:y
请输入一个表达式:
1+(10+2-1+2)
原表达式:1+(10+2-1+2)
逆波兰式: 1 10 2 +1 -2 ++
计算机结果:22
continue y or n?:
```

3、实验总结

通过本次实验，了解了有穷状态自动机的概念，存储。如何建立有穷状态自动机与正则文法之间的联系。了解了前中后缀表达式以及逆波兰表达式，知道了计算机中如何进行运算，通过后缀表达式实现逆波兰计算器。在实验中，我的编程水平也进一步提高，对程序的编写也更加熟练。