编译原理实验报告

基于 LL（1）的语法分析器

161250155 吴林漾

实验目的

构造一个自定义语法分析程序，实现语法分析器，基于 LL(1)语法分析方法对输入语句进行分析，并输出结果。

内容描述

此程序用 java 编写。程序读取一个文本文件，并对其中的序列进行语法分析。使用 LL(1)方法自顶向下进行分析，输出分析过程中的匹配情况和产生式序列。
思路方法
1.自定义文法 G
2.对 G 进行预处理，消除左递归、二义性，形成文法 G'
3.计算所有非终结符的 First、Follow
4.构造 Prediction Parsing Table
5.根据输入队列和状态栈的栈顶元素进行分析，将终结符进行匹配或非终结符产生子项，循环处理至输入队列队尾
6. 输出匹配情况和产生式序列

假设

1.输入的序列仅含有 i, +, -, *, /, (,)符号

相关分析过程描述

自定义文法

G:
E-> E+T | E-T | T
T-> T*F | T/F | F
F-> (E) | T

文法预处理

G':
E-> TC
C-> AC | ε
a-> +T | -T
T-> FD
D-> BD | ε
B-> *F | /F
F-> (E) | i

计算非终结符的 First、Follow 集

	First	Follow
E	{ (, i }	{+, -, ε}
T	{ (, i }	{+, -,), $ }

F	{ (, i}	{+, -, *, /,), $}
A	{+, -}	{+, -,), $}
B	{+, -, *, /,), $}	{+, -, *, /,), $}
C	{+, -, ε}	{+, -, *, /,), $}
D	{*, /, ε}	{+, -,), $}

构造 PPT

	+	-	*	/	(	)	i	$
E					E->TC		E->TC	
C	C->AC	C->AC				C->ε	C->ε	
A	A->+T	A->-T						
T					T->FD		T->FD	
D	D->ε	D->ε	D->BD	D->BD		D->ε		D->ε
B			B->*F	B->/F				
F					F->(E)		F->i	

数据结构定义

PPT 格式如下：

```
public class ParsingTable {

    static String[][] table= {{"null", "null", "null","null","E->TC","null","E->TC"},
                              {"C->AC","C->AC","null","null","null","C->#","null"},
                              {"A->+T","A->-T","null","null","null","null","null"},
                              {"null","null","null","null","T->FD","null","T->FD"},
                              {"D->#","D->#","D->BD","D->BD","null","D->#","null"},
                              {"null","null","B->*F","B->/F","null","null","null"},
                              {"null","null","null","null","F->(E)","null","F->i"}};

    public ParsingTable(){
    }

    public String getExpression(int i,int j){
        return table[i][j];
    }
}
```

输出格式：

List<String>

核心算法

```
public LLParser() {
    output=new ArrayList<>();
    identifier=new Stack<>();
    identifier.push( item: 'E');
    table=new ParsingTable();
}

public void analyze(String input){
    int i=0;
    while (input.charAt(i)!='$'){
        char x=input.charAt(i);
        char y=identifier.peek();
        if (x==y){
            identifier.pop();
            output.add("match: "+x);
            i++;
            continue;
        }
        else if (table.getExpression(getIndex(y),getIndex(x)).equals("null")){
            output.add("error");
            break;
        }else{
            String expr=table.getExpression(getIndex(y),getIndex(x));
            output.add(expr);
            String id=expr.substring(expr.indexOf(">")+1);
            identifier.pop();
            for (int j=id.length()-1;j>=0;j--){
                if (id.charAt(j)!='#')
                    identifier.push(id.charAt(j));
            }
            continue;
        }
    }
}
```

程序有状态栈和输入队列，在输入字符序列最后加上终止符，放进队列，状态栈中压入非终结符。读取栈和队列的第一个元素分析，如果匹配成功，则弹出，如果不匹配，则查询 PPT 表的相关产生式，把新元素压栈，循环至队尾。如果过程中终结符不匹配，则输出 error。

运行截图

输入内容如下：

```
i+i-i*i+i/(i+i)-i
```

输出内容如下：

```
E->TC
T->FD
F->i
match: i
D->#
C->AC
A->+T
match: +
T->FD
F->i
match: i
D->#
C->AC
A->-T
match: -
T->FD
F->i
match: i
D->BD
B->*F
match: *
F->i
match: i
D->#
C->AC
A->+T
match: +
T->FD
F->i
match: i
D->BD
B->/F
match: /
F->(E)
match: (
E->TC
T->FD
```

```
F->i
match: i
D->#
C->#
match: )
D->#
C->AC
A->-T
match: -
T->FD
F->i
match: i
```

问题与解决

1. PPT 表出现了数组越界的问题，后经修改得以解决
2. IOHandler 出现了文件输入的问题，经过修改输入流形式解决

感想

经过自己动手查资料、编写语法分析程序，有助于对语法分析过程和方法有更深入的理解。