上海大学

SHANGHAI UNIVERSITY

<编译原理>实验报告

学　　院　计算机工程与科学学院

组　　号　8

实 验 题 号　三

日　　期　2022 年 5 月 6 日

学号	姓名	主要工作	贡献因子
16121803	许睿	代码调试	20%
19120188	孙瑶	代码测试	20%
19120191	汪雨卿	代码优化	20%
19121442	曹卓文	代码编写	20%
19122408	严邹莹	报告撰写	20%

实验三 语法分析

一、实验目的与要求

1、给出 PL/0 文法规范，要求编写 PL/0 语言的语法分析程序。

2、通过设计、编制、调试一个典型的语法分析程序，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，进一步掌握常用的语法分析方法。

3、选择一种语法分析方法（递归子程序法、LL(1)分析法、算符优先分析法、SLR(1)分析法）；选择常见程序语言都具备的语法结构，如赋值语句，特别是表达式，作为分析对象。

二、实验环境

c/c++、visual studio

三、实验内容

1、已给 PL/0 语言文法，构造表达式部分的语法分析器。

2、分析对象〈算术表达式〉的 BNF 定义如下：

<表达式> ::= [+|-]<项>{<加法运算符> <项>}

<项> ::= <因子>{<乘法运算符> <因子>}

<因子> ::= <标识符>|<无符号整数>| ‘(’<表达式>‘)’

<加法运算符> ::= +|-

<乘法运算符> ::= *|/

四、实验内容的设计与实现

4.1 实验设计

利用了预测分析法的思想，通过建立预测分析表和符号栈从而编写预测分析

程序。

预测分析程序基于 LL(1)文法，文法如下所示：

e -> PE | E

E -> TE'

E' -> PTE' | ε

T -> FT'

T' -> MFT' | ε

F -> I | (e)

I -> i | n

P -> + | -

M -> * | /

根据该文法求出每个表达式的 SELECT 集，结果如下所示：

SELECT(e->PE) = {+, -}

SELECT(e->E) = {i, n, (}

SELECT(E->TE') = {i, n, (}

SELECT(E'->PTE') = {+, -}

SELECT(E'->ε) = FOLLOW(E') = {) , #}

SELECT(T->FT') = {i, n, (}

SELECT(T'->MFT') = {*, /}

SELECT(T'->ε) = FOLLOW(T') = {+, -,), #}

SELECT(F->I) = {i, n}

SELECT(F->(e)) = { (}

SELECT(I->i) = {i}

SELECT(I->n) = {n}

SELECT(P->+) = {+}

SELECT(P->-) = {-}

SELECT(M->*) = {*}

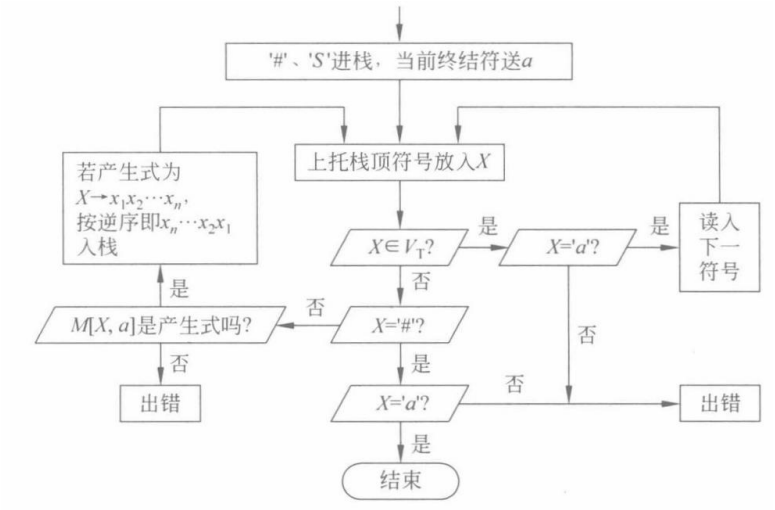
SELECT(M->/) = {/}

写出预测分析表如下所示：

终结符 / 非终结符	i	n	+	-	*	/	(	)	#
e	→E	→E	→PE	→PE			→E		
E	→TE'	→TE'					→TE'		
E'			→PTE'	→PTE'				→ε	→ε
T	→FT'	→FT'					→FT'		
T'			→ε	→ε	→MFT'	→MFT'		→ε	→ε
F	→I	→I					→(e)		
I	→i	→n							
P			→+	→-					
M					→*	→/			

表中所有空白位置均表示该产生式不可到达，为出错状况。

算法流程图如下所示：



4.2 主要代码实现

1、预测分析表

```
const std::map<Element, std::map<Element, std::vector<Element>>> prediction_table
= {
        {expr_,
                {{plus_, {Plus,     Expr}},
                 {minus_, {Plus,     Expr}},
                 {ident,    {Expr}},
                 {number, {Expr}},
                 {lparen, {Expr}}
                }
        },
        {Expr,
                {{ident, {Term,     expr}},
                 {number, {Term,     expr}},
                 {lparen, {Term,     expr}}
                }
        },
        {expr,
                {{plus_, {Plus,     Term, expr}},
                 {minus_, {Plus,     Term, expr}},
                 {rparen, {}},
                 {null,     {}}
                }
        },
```

```
{Term,
        {{ident, {Factor, term}},
         {number, {Factor, term}},
         {lparen, {Factor, term}}
        }
},
{term,
        {{plus_, {}},
         {minus_, {}},
         {times,   {Multi,   Factor, term}},
         {slash,   {Multi, Factor, term}},
         {rparen, {}},
         {null, {}}
        }
},
{Factor,
        {{ident, {Ident}},
         {number, {Ident}},
         {lparen, {lparen, expr_,   rparen}}
        }
},
{Plus,
        {{plus_, {plus_}},
         {minus_, {minus_}}
```

```
                }
        },
        {Multi,
                {{times, {times}},
                 {slash,   {slash}}
                }
        },
        {Ident,
                {{ident, {ident}},
                 {number, {number}}
                }
        }};
```

2、符号栈处理

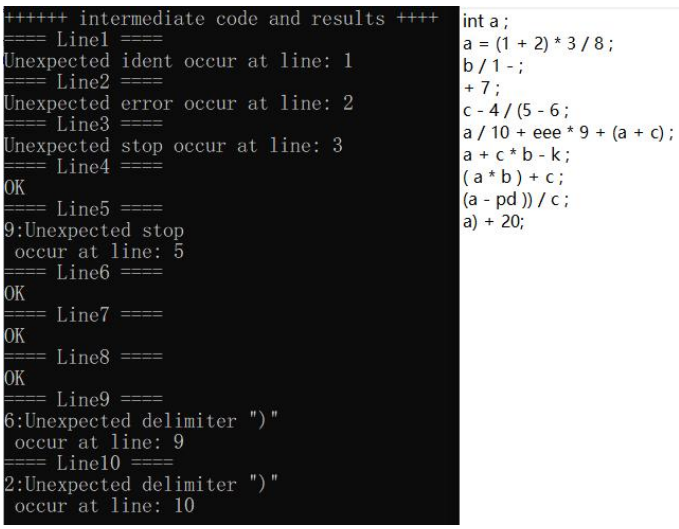
```
// 如果不是终结符则循环
        while (s.top()>=expr_) {
            auto cur = s.top(); // 拿栈顶
            s.pop(); // 出栈
            auto t = next(cur, token.first); // cur->非终结符  token.first->当前token 的类型
            if (!t.empty() && t.at(0) == error)return false; // 错误处理
            for (auto &it: t) s.push(it); // 入栈
        }
        if (s.top() != token.first) { // 如果 token 类型和顶部（此时为终结符）不
```

```
匹配，出错
                std::cout << counter << ':';
                std::cout << get_error_type(token.first) << '\n';
                return false;
            }
            s.pop();
```

4.3 实验结果

左图为测试数据，右图为输出结果。

数据一：



该数据包含了 10 个测试样例，根据运行结果可以看到，第 1 个测试样例出现了两个连续的标识符；第 2 个测试样例出现了不可接受的"="符，第 3 个测试样例出现了意外终止的"-"符；第 5 个测试样例同样也是出现了意外终止导致括号不匹配；第 9 个和第 10 个测试样例都是出现了多余的右括号，导致括号不匹配。

五、收获与体会

许睿：

通过前期课程的学习，我对编译器对程序语言的分析过程有了比较详细的理解，但是如何编程实现这些功能方面掌握不足。在实验的过程中，总觉得书上附录相关的词法分析器，语法分析器等的相关伪程序，内容不足以支撑整个实验。在于其他同学的交流中，一同查阅了预测分析法的补充资料，让我对课程的理解更加深刻。

孙瑶：

本次实验在语法分析过程中根据 SSL 文法进行栈的建立，并且在具体分析中先定义了各类表达式的 BNF 定义，以便判断语法正确性的循环能够顺利进行，而在测试用例的编写过程中尽可能考虑多种出错情况，从而使代码更优，在整个实验过程中让我对语法分析有了更形象深入的理解。

汪雨卿：

本次实验需要完成的是语法分析，通过讨论我们决定使用建立预测表驱动的 LL(1)分析程序。在实现和优化代码的过程中，我更深入了解了语法分析部分的处理过程，求算 FIRST 和 FOLLOW 集，然后推导出预测表，以及读入字符串之后的移进和规约操作。一点疏漏都会影响到整体的分析。由于语法分析过程是建立在之前词法分析的基础上，之前词法分析过程中新出现的末字符读入问题，也花费了我们一些时间去修复。

曹卓文：

本次实验的主要内容是语法分析，在实验过程中，我们首先建立了 LL(1)文法，接着，通过文法构造了预测分析表，在代码实现过程中，通过栈和映射表实现了最左推导，在此过程中对符号栈如何正确处理，如进出栈顺序等问题有了深入的理解。尤其是通过本次实验的实践，让我对预测分析法有了更好的了解。

严邹莹：

此次实验是进行语法分析，我们通过预测分析法的思想建立符号栈和预测分析表，由于该方法对于文法的要求较高，在文法的构造和化简上也经历了一些困难。此外，在编写代码的过程中，也深刻地体会到了符号栈进栈顺序的重要性，让我对于预测分析法以及其他语法分析方法有了进一步的理解。