東南大學
SOUTHEAST UNIVERSITY

编译原理实验

报告二

实验名称： Syntax Parser

学生姓名： 白丰硕

学生学号： 71116233

东南大学计算机科学与工程学院、软件学院

School of Computer Science & Engineering College of Software Engineering

Southeast University

二 0 一 九 年 一 月

東南大學
SOUTHEAST UNIVERSITY

目录

一、 实验目的

(1) 巩固下推自动机理论与自上而下的语法分析，两者之间的关系与相关的知识点

(2) 了解语法分析程序具体的实现方法及所涉及算法的原理

(3) 编写相关的语法分析程序实现对 C++语言程序文件的语法分析

二、 实验环境

(1) 开发环境：

- OS：Ubuntu 16.04
- 语言：Python 3.6
- 画图：ProcessOn
- 打包：Pyinstaller

(2) 运行环境

- Windows 或 Linux

三、 实验内容

编写代码实现一个语法分析程序，可以对 C++语言的.cpp 和.h 文件中的语句进行语句语法分析，在词法分析的基础上，对词法分析产生的 token 序列进行进一步的处理与分析，通过采用自上而下的分析方法，构造产生式并构建预测分析表，编写实现下推自动机，实现语法分析过程，打印下推自动机推导序列。

主要功能

➢ 读写文件

语法分析程序可以从指定的文件路径下，读取指定的文件，先进行词法分析过程，并将做好词法分析的 token 流存储到队列中，并进行语法分析，读取预测分析表以及产生式表，通过比对查表获得产生式序号。

➢ 语法分析

语法分析主程序将处理好的 token 队列开始进行逐个扫描，进行相应的语法分析。比对栈顶内容与读头下内容，再通过查询预测分析表和产生式表获得一步推导，将推导结果压到栈中，并重复这一过程直到出现错误或者完全推导结束

➢ 外部参数

本程序中提供外部参数，使得语法分析程序可以更加灵活的分析、运行程序，可以选择任何路径下的文件作为被分析文件，并将生成的 derivation 序列文件放置于本目录下的 derivation 文件夹中

参数	描述
-h	show this help message and exit
-f，--file	path of the file to analyze

单词表

➢ basicWordTable

"include","define","auto","bool","break","case","catch","char","class",

"const","const_cast","continue","default","delete","do","double",

"dynamic_cast","else","enum","explicit","extern","false","float","for",

"friend","goto","if","inline","int","long","mutable","namespace","new",

"operator","private","protected","public","register","reinterpret_cast",

"return","short","signed","sizeof","static","static_cast","struct",

"switch","template","this","throw","true","try","typedef","typeid",

"typename","union","unsigned","using","virtual","void","volatile","while"

- **separatorTable**

",",",",";",".","(",")","[","]","{","}","#"

- **operatorTable**

"+", "-","++", "--", "*", "/", "<", "<=", ">", ">=", "=", "==","!="

四、 实验结果

该实验结果由以下这个简短的 C++程序所构造得出，只是为了展示语法分析效果，更多的示例在 sample 文件下可以找到。

```
include<iostream>
float inc(float q){
    return q++;
}
/*
这是一个没什么用的注释
*/
int main(){
    //声明一个变量
    int a = inc(1.2);
}
```

```
reader:#include<
stackPointer:P
product:P ->PC DL
stack:['#', 'DL', 'PC']

reader:#include<
stackPointer:PC
product:PC->HL NL
stack:['#', 'DL', 'NL', 'HL']

reader:#include<
stackPointer:HL
product:HL->#include< Id > H1
stack:['#', 'DL', 'NL', 'H1', '>', 'Id', '#include<']

reader:#include<
stackPointer:#include<
success!
stack:['#', 'DL', 'NL', 'H1', '>', 'Id']

reader:Id
stackPointer:Id
success!
stack:['#', 'DL', 'NL', 'H1', '>']

```

五、 数据结构

名称	类型	描述
queue	Queue	用于放置 token 序列
stack	Stack	下推栈
ppt	Sheet	用于存放预测分析表
product	Sheet	用于存放产生式序号对应表

六、 实验算法

本次实验中语法分析主要采用的是自上而下的语法分析，LL（1）来分析程序。通过构造产生式表和预测分析表，来就行推导式序列的生成。

➢ 文法 G

Id	推导式	说明
1	P ->PC DL	程序推出前置代码和声明列表
2	PC->HL NL	前置代码推出头文件列表或者名字空间列表
3	HL->HL #include<Id>| HL #include”Id”| #include<Id>| #include”Id”	头文件列表推出头文件列表和头文件或头文件
4	NL->NL using namespace Id| using namespace Id	名字空间列表推出名字空间列表和名字空间或者名字空间
5	DL->DL D|D	声明列表推出声明列表和声明或者声明
6	D ->Dv|Df	声明推出变量声明或者程序声明
7	Dv->T Id	变量声明推出类型符号和标识符
8	T -> int|float|double|char|void	类型符号推出常用的类型符
9	Df->T Id(Pm) CS	函数声明推出类型、函数名标识符、参数列表和代码块
10	Pm->PL|void	函数参数推出参数列表或者 void
11	PL->PL,p|p	参数列表推出参数列表，参数或者参数
12	p->T Id	参数推出类型符号和标识符
13	CS->{Dl SL}	函数体推出局部变量声明和语句列表
14	Dl->Dl Dv|E	局部变量声明推出局部变量声明和变量声明或者表达式
15	SL->SL S|E	语句列表推出语句列表和语句或者表达式
16	S->ES|CS|SS|IS|RS	语句推出表达式语句|代码块|选择语句|迭代语句|返回语句
17	ES->E;	表达式语句推出表达式和分号
18	SS->if (E) S|if (E) S else S	选择语句推出 if…或者 if…else…
19	IS->while (E) S	迭代语句推出 while…
20	RS ->return;|return E	返回语句推出 return 空或者表达式
21	E->Id=E|Es	表达式推出标识符赋值或者简单表达式
22	Es->Ea relop Ea|Ea	简单表达式推出两个表达式的 r 运算或者表达式
23	relop-><= |<|>|>=|==|!=|&&| ||	推出比较运算符和逻辑算术运算符
24	Ea->Ea addop Eb|Eb	表达式 a 推出表达式 ab 之间的 a 运算或者表达式 b
25	addop->+|-	a 操作推出加减
26	Eb->Eb mulop Ec | Ec	表达式 b 推出表达式 bc 之间的 m 运算或者表

		达式 c
27	mulop->* | /	m 操作推出乘除
28	Ec->(E) | Id | NUM	表达式 c 推出带括号表达式或者变量或常量

➢ 预处理

预处理过程主要是包含两个部分，消除左递归和提取最大公共左因子。从而获得通过预处理的文法G'

➢ 文法G'

Id	推导式	说明
1	P ->PC DL	程序推出前置代码和声明列表
2	PC->HL NL	前置代码推出头文件列表或者名字空间列表
3	HL->#include<Id> H1	头文件列表推出头文件和 H1
4	HL->#include”Id”H1	
5	H1->HL	提取公共左因子
6	H1-> ε	
7	NL->using namespace Id N1	名字空间列表推出名字空间和 N1
8	N1->NL	提取公共左因子
9	N1-> ε	
10	DL->D DO	声明列表推出声明和 D1
11	DO->DL	提取公共左因子
12	DO-> ε	
13	D ->T Id Dt	声明推出变量声明或者程序声明
14	Dt->Df	
15	Dt->;	
16	T -> int	类型符号推出常用的类型符
17	T -> float	
18	T -> double	
19	T -> char	
20	T -> void	
21	Df->(Pm) CS	函数声明推出类型、函数名标识符、参数列表和代码块
22	Pm->PL	函数参数推出参数列表
72	Pm-> ε	
23	PL->p P1	参数列表推出参数和 P1
24	P1->,PL	提取公共左因子
25	P1-> ε	
26	p->T Id	参数推出类型符号和标识符

27	CS->{Dl SL}	函数体推出局部变量声明和语句列表
28	Dl->T Dl	局部变量声明推出变量声明和局部变量声明或者表达式
29	Dl->E	
30	SL->S Sp	语句列表推出语句和语句列表或者表达式
31	Sp->SL	
32	Sp-> ε	
33	S->ES	语句推出表达式语句|代码块|选择语句|迭代语句|返回语句
34	S->CS	
35	S->SS	
36	S->IS	
37	S->RS	
38	ES->E;	表达式语句推出表达式和分号
39	SS->if (E) S S1	选择语句推出 if…S…S1
40	S1->else S	提取公共左因子
41	S1-> ε	
42	IS->while (E) S	迭代语句推出 while…
43	RS ->return R1	返回语句推出 return 和 R1
44	R1->;	提取公共左因子
45	R1->E	
46	E->Id=E;	表达式推出标识符赋值或者简单表达式
47	E->Es	
48	Es->Ea Es1	简单表达式推出表达式 a 和 Es1 达式
49	Es1->relop Ea	提取公共左因子
50	Es1-> ε	
51	relop-><=	推出比较运算符和逻辑算术运算符
52	relop-><	
53	relop->>	
54	relop->>=	
55	relop->==	
56	relop->!=	
57	relop->&&	
58	relop->||	
59	Ea-> Eb Ea1	表达式 a 推出表达式 b 和 Ea1
60	Ea1->addop Eb Ea1	消除左递归
61	Ea1-> ε	
62	addop->+	a 操作推出加减
63	addop->-	
64	Eb->Ec Eb1	表达式 b 推出表达式 c 和 Eb1
65	Eb1->mulop Ec Eb1	消除左递归
66	Eb1-> ε	
67	mulop->*	m 操作推出乘除
68	mulop->/	

69	Ec->(E)	表达式 c 推出带括号表达式或者变量或常量
70	Ec->Id	
71	Ec->NUM	

➢ First

V_N	First
P	#include“,#include<
PC	#include“,#include<
DL	int,float,double,char,void
HL	#include“,#include<
NL	using namespace
H1	#include“,#include<,ε
N1	using namespace,ε
D	int,float,double,char,void
Dt	(,;
D0	int,float,double,char,void,ε
Df	(
T	int,float,double,char,void
Pm	int,float,double,char,void
CS	{
PL	int,float,double,char,void
p	int,float,double,char,void
P1	,|ε
Dl	int,float,double,char,void,Id,(,NUM
SL	(,Id,NUM,{,if,while,return
E	(,Id,NUM
S	(,Id,NUM,{,if,while,return
Sp	(,Id,NUM,{,if,while,return,ε
ES	I(,Id,NUM
SS	if
IS	while
RS	return
S1	else,ε
R1	;, (,Id,NUM
Es	(,Id,NUM
Es1	<= ,<,>,>=,==,!=,&&,||,ε
relop	<= ,<,>,>=,==,!=,&&,||
Ea	(,Id,NUM
Ea1	+,-,ε
addop	+,-
Eb	(,Id,NUM
Eb1	*,/,ε

mulop	*,/
Ec	(,Id,NUM

- **Follow**

V_N	Follow
H1	using namespace
N1	int,float,double,char,void
D0	#
P1	)
S1	(,Id,NUM,{,if,while,return,else
Sp	}
Es1	(,Id,NUM,{,if,while,return,else,},),;
Ea1	(,Id,NUM,{,if,while,return,else,},),;,<= ,<,>,>=,==,!=,&&,||
Eb1	(,Id,NUM,{,if,while,return,else,},),;,<= ,<,>,>=,==,!=,&&,||,+,-

- **构造 PPT**

通过 First 集合与 Follow 集合构造 PPT，用以对语法进行分析并产生相应所需要的推导式序列。

七、 实验心得

在本次实验中，通过个人使用 python，编写了 syntax.py 实现了 C++语法分析器。在本次实验中，我主要是参考了网上的一篇博客[1]，使用了 BNF，但是产生式都是本人经过思考构造得出的。在产生式构造的方面，由于产生过于繁多，在消除左递归和提取公因子遇到不少麻烦。由于使用的自上而下的分析方法，LL(1)的分析能力是很有限的，二义文法是处理不了的，只能对 ppt 做出强行的规定，所以对于可以分析的语法就受到一定的限制。语法分析比较局限，但是分析的内容相对还是比较复杂，目前一共有 70+条产生式，可以对基本的头文件声明、名字空间使用声明、函数声明、变量初始化、分支语句、循环语句、返回语句、运算表达式（加减乘除）、表达式比较（关系云算法）等都可以进行分析处理。本次程序中，让我更加深刻的体会到语法分析的具体过程与自上而下的分析方法的具体细节。本次实验中所编写的语法分析器虽初具 C++语法分析能力，但是还是有很多的欠缺与不足，以后有更多的时间会进一步的弥补不足与缺陷。

八、 参考内容

[1] https://blog.csdn.net/pinbodexiaozhu/article/details/25394417

C—的 BNF 语法