编译原理实验报告

类 C 语言语法分析器

学　　号　1750844

姓　　名　周展田

专　　业　计算机科学与技术

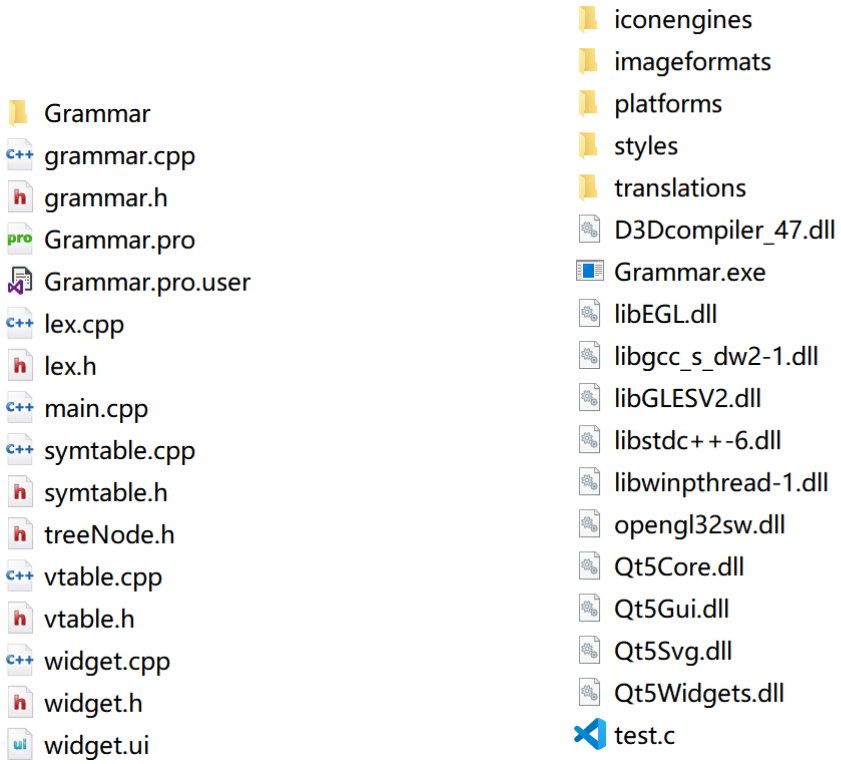
授课老师　高秀芬

1.功能描述

使用 LL(1)进行分析的类 C 语言语法分析器(文法 2_不包含过程调用)。

2.文件说明

本次实验使用 Qt 完成，目录中包含了程序的源文件和工程文件，Grammar 文件夹中含有可执行程序 Grammar.exe。主要的词法分析由 Lex 类完成，语法分析由 Grammar 和 VTable 类完成，在 widget 类中进行图形界面设计并生成词法分析和语法分析对象。



3.设计思路

(1)化简文法，使其符合 LL(1)规则

Program ::= <类型> < ID>’(‘ ‘)’ <语句块>

<类型>::=int | void

<语句块> ::= ‘{ ‘<内部声明> <语句串>’ }’

<内部声明> ::= 空 | <内部变量声明><内部声明>

<内部变量声明>::=int <ID><next>;

<next> ::= ,<ID><next> | 空

<语句串> ::= <语句> <语句串> | 空

<语句> ::= <if 语句> |< while 语句> | <return 语句>; | <赋值语句>;

<赋值语句> ::= <ID> =<表达式>

<return 语句> ::= return retBlock

retBlock ::= 空 | 表达式

<while 语句> ::= while '(' <表达式> ')' <语句块>

<if 语句> ::= if '(' <表达式> ')' <语句块> elseBlock

elseBlock ::= else <语句块> | 空

<表达式>::=<加法表达式> <comp>

<comp> ::= relop <加法表达式><comp> | 空

<加法表达式> ::= <项> <op1>

<op1> ::= +<项><op1> | 空

<项> ::= <因子> <op2>

<op2> ::= * <因子> <op2> | 空

<因子> ::=num | '(' <表达式> ')' |<ID>

(2)用定义好的枚举类型表示产生式

typedef enum { INT, VOID, ID, LP, RP, LB, RB, WHILE, IF, ELSE, RETURN, ASSIGN, OP1, OP2, RELOP, DEL, SEP, NUM, NL, PROGRAM, TYPE, SENBLOCK, INNERDEF, INNERVARIDEF, NEXT, SENSEQ, SENTENCE,

ASSIGNMENT, RETURNSEN, RETBLOCK, WHILESEN, IFSEN, ELSEBOCLK, EXPRESSION, COMP, PLUSEX, OPPLUSDEC, TERM, OPMULDIV, FACTOR, EPSILON, END, ERROR}tokenType;

(3)符号表

符号表	Enum
int	INT
void	VOID
ID	ID
(	LP
)	RP
{	LB
}	RB
while	WHILE
if	IF
else	ELSE
return	RETURN
=	ASSIGN
+ -	OP1
* /	OP2
> < >= <= == !=	RELOP
;	DEL
,	SEP
num	NUM
\n	NL
<Program>	PROGRAM
<类型>	TYPE
<语句块>	SENBLOCK
<内部声明>	INNEREDEF
<内部变量声明>	INNERVARIDEF
<next>	NEXT
<语句串>	SENSEQ
<语句>	SENTENCE
<赋值语句>	ASSIGNMENT
<return 语句>	RETURNSEN
<retBlock>	RETBLOCK
<while 语句>	WHILESEN
<if 语句>	IFSEN
<elseblock>	ELSEBLOCK
<表达式>	EXPRESSION

<comp>	COMP
<加法表达式>	PLUSEX
<op1>	OPPLUSEDEC
<项>	TERM
<op2>	OPMULDIV
<因子>	FACTOR
空	EPSILON
#	END

(4)表达式生成

```
class sentence//用来表示右部生成式
{
public:
    sentence* next;
    tokenTypes tokens[6];
    int tokenLen;
};

class generator//用来表示左部的非终结符
{
public:
    tokenTypes head;
    sentence* descri;
};
```

(5)构建 FIRST 和 FOLLOW 集合

PROGRAM=: INT:TYPE ID LP RP SENBLOCK | VOID:TYPE ID LP RP SENBLOCK

TYPE=: INT:INT | VOID:VOID

SENBLOCK=: LB:LB INNERDEF SENSEQ RB

INNERDEF=: INT:INNERVARIDEF INNERDEF | ID:EPSILON | RB:EPSILON |

IF:EPSILON | ELSE:EPSILON | RETURN:EPSILON

INNERVARIDEF=: INT:INT ID NEXT DEL

NEXT=: DEL:EPSILON | SEP:SEP ID NEXT

SENSEQ=: ID:SENTENCE SENSEQ | RB:EPSILON | WHILE:SENTENCE SENSEQ |

IF:SENTENCE SENSEQ | RETURN:SENTENCE SENSEQ

SENTENCE=: ID:ASSIGNMENT DEL | WHILE:WHILESEN | IF:IFSEN | RETURN:RETURNSEN DEL

ASSIGNMENT=: ID:ID ASSIGN EXPRESSION

RETURNSEN=: RETURN:RETURN RETBLOCK

RETBLOCK=: ID:EXPRESSION | LP:EXPRESSION | DEL:EPSILON | NUM:EXPRESSION

WHILESEN=: WHILE:WHILE LP EXPRESSION RP SENBLOCK

IFSEN=: IF:IF LP EXPRESSION RP SENBLOCK ELSEBOCLK

ELSEBOCLK=: ID:EPSILON | RB:EPSILON | WHILE:EPSILON | IF:EPSILON | ELSE:ELSE SENBLOCK | RETURN:EPSILON

EXPRESSION=: ID:PLUSEX COMP | LP:PLUSEX COMP | NUM:PLUSEX COMP

COMP=: RP:EPSILON | RELOP:RELOP PLUSEX COMP | DEL:EPSILON

PLUSEX=: ID:TERM OPPLUSDEC | LP:TERM OPPLUSDEC | NUM:TERM OPPLUSDEC

OPPLUSDEC=: RP:EPSILON | RETURN:EPSILON | OP1:OP1 TERM OPPLUSDEC | RELOP:EPSILON | DEL:EPSILON

TERM=: ID:FACTOR OPMULDIV | LP:FACTOR OPMULDIV | NUM:FACTOR OPMULDIV

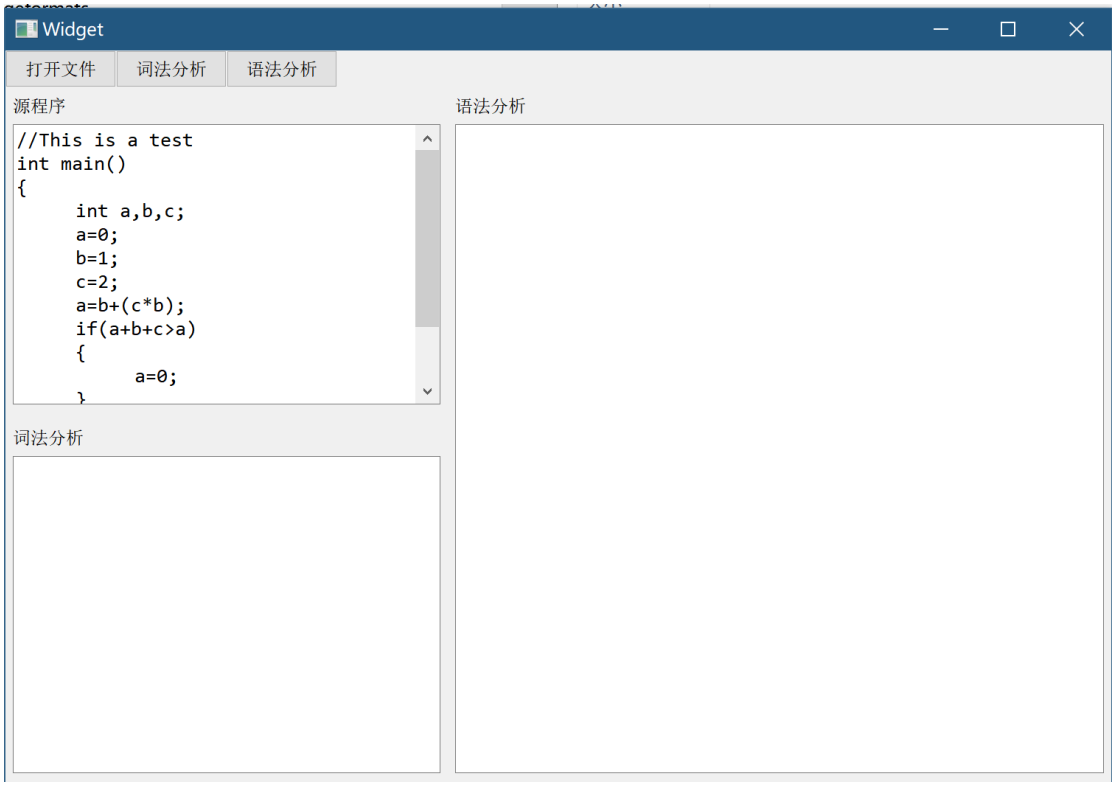
OPMULDIV=: RP:EPSILON | RETURN:EPSILON | OP1:EPSILON | OP2:OP2 FACTOR OPMULDIV | RELOP:EPSILON | DEL:EPSILON

FACTOR=: ID:ID | LP:LP EXPRESSION RP | NUM:NUM

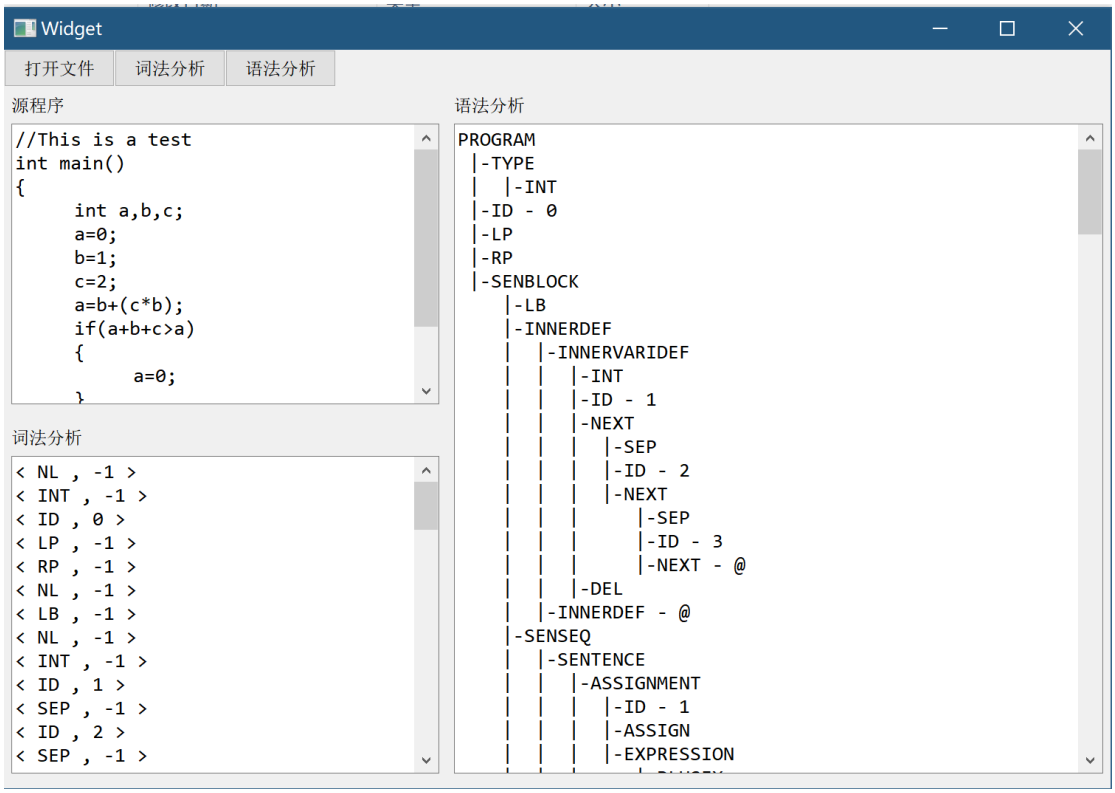
(6)构造 LL(1)分析表

4.结果演示

打开测试文件：



可以先进行词法分析，再进行语法分析；也可以直接打开后进行语法分析，此时文法分析将会被作为子程序调用：



词法分析的结果被保存到 Lex.txt 中，符号表保存在 symbol.txt 中，语法分析结果保存在 Gram.txt 中。