南开大学
编译原理实验报告
语法分析器

专业：软件工程

组名：贯彻习大大重要回信精神小组

成员：公岩松（1511419）

成员：郭迎港（1511423）

成员：孟宇航（1511443）

任课教师：马玲

2017 年 11 月

一、实验描述:

语法分析是编译器构造的重要环节之一，为了巩固课堂知识，加强对语法分析的理解，我们将完对语法分析器的实际编写，最终生成抽象语法树，为接下来的工作做好铺垫。

二、实验目的及要求:

目的： 1.熟悉课堂知识

2.完成语法分析器

3.生成抽象语法树

要求： 1.使用 c/c++进行编程

2.完成语法分析器所需基本功能

3.输出生成的语法树结构

三、实验环境:

平　台：windows

编译器：visual stdio 2013

工　具：Flex/Bision

四、成员分工:

公岩松：语法树数据结构维护、符号表维护、类型系统构建等；
郭迎港：文法设计、语义动作编写等；
孟宇航：常量表达式计算、测试、报告撰写等。

五、实验步骤:

1.产生式的构建(详见附录):

我们使用了 C 语言的标准文法，其中通过分级的产生式来解决运算符等的优先级和结合性问题，通过为这些文法添加语义动作，我们最终支持的语法如下：所有基本的数据类型、语句和运算（包括++、--），以及复合语句、函数（支持作用域）、注释、输入输出语句、跳转语句、数组、指针、自动类型 auto、sizeof、强制类型转换等。

2.语法树的构建:

我们采用二叉树的方式构建语法树，表示方式为左孩子右兄弟表示法，其主要优势如下：

（1） 树形结构并不容易维护，而二叉树则很容易维护，而且非常稳定。

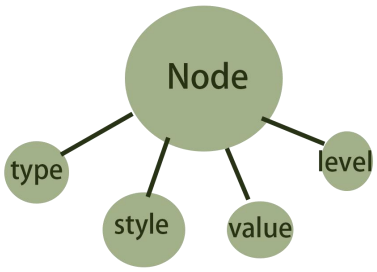
（2） 左孩子右兄弟表示法只需要先序遍历就可以完成目标格式的输出。

（3） 应用自身写的库（复用数据结构课程中完成的代码）具有非常强的可定制性，可以从底层完成我们需要的功能。

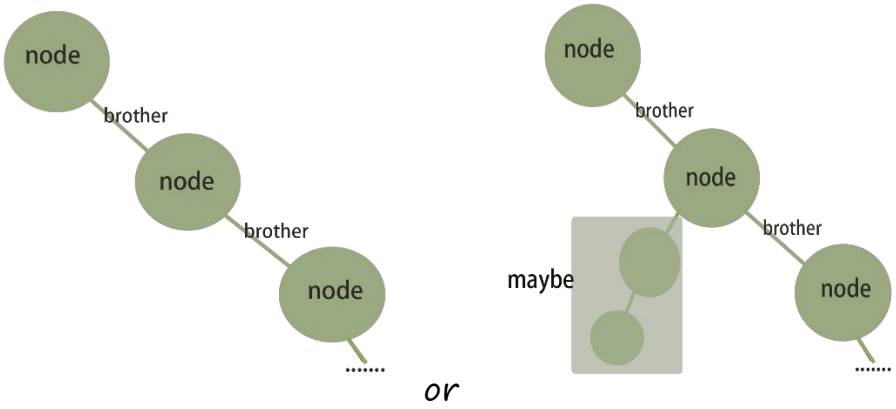
（4） 自底向上建树非常的容易实现。

我们构建语法树的过程如下：

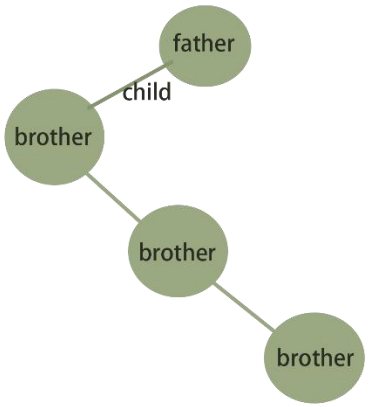
（1）对于每一条产生式，首先构建自己的节点:



（2）产生式右端每个终结符/非终结符看做同级，兄弟节点连接:



（3）将产生式左端非终结符作为父节点，构成一个完整的树:



（4）根据文法自底向上最终构建整个语法树

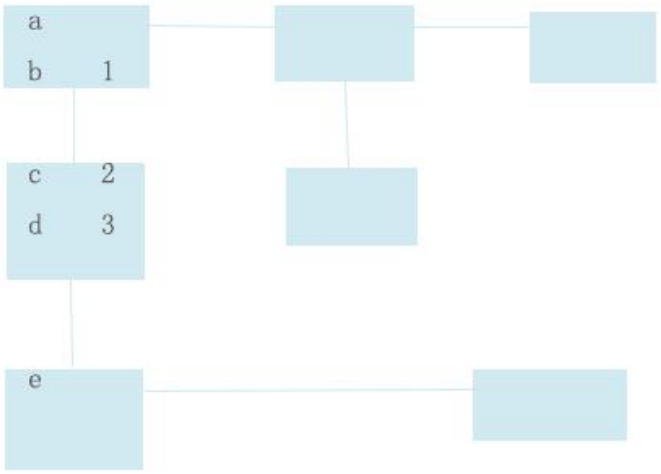
3.符号表的构建:

这次符号表是在原有的基础上的拓展，我们上次是以链表连接的哈希表作为符号表，这种符号表可以完成绝大部分的功能，解决基础的作用域问题。然而遇到一些如结构体之类的带名字的作用域，这样的方式就出现问题了。如果按照常理来看，需要一个树形结构节点为哈希表来储存符号。然而这样做会出现这样的问题，我们需要遍历兄弟节点，并且判断每个兄弟节点是否符合我们寻找的节点，这样效率非常低，而且很难在原有基础上实现。而且还有许多的寻找父节点，这种操作，也很难在原有基础上改写。

我们在原有的基础上在没一个符号中加入了符号表的指针，也就是说我们用链表将链哈希表连接起来，这样子仿造构成一个树形结构，根据原来的机制，没有名字的作用域是访问不到的，会自动弹出，其他任何作用域都可以根据指针访问到，而且由于有两层链表，把多层作用域出现的问题解决了。而且值得一提的是这样构建的符号表的寻找符号的时间复杂度非常低，由于哈希表的搜寻是接近O(1)的，也就是说我们最多时间复杂度是这两层链表的总深度，然而一个程序

几乎不可能会超过十万个以上的作用域，这样来看，我们搜索符号的速度是非常快的，相对而言，我们牺牲了一部分空间复杂度。所以当作用域没有名字时，就（几乎）代表着我们之后无法访问到这个作用域了，当出这个作用域时会自动弹出。

当访问符号表时，我们根据符号表中的当前表指针与指针栈，可以逆着寻找到任何符号表的父符号表，这样就可以解决绝大部分的作用域问题。



4.类型系统的构建

由于追求构建的速度，所以这次的类型系统牺牲了可拓展性，从而造成了一些问题，也有很多问题无法用此系统解决。

类型系统的构建思路如下：

使用宏将所有类型定义为整数，包括 int, char, float, double, pointer, array 之后根据整数的整除和取模运算可以得到这个类型的一些信息，如数组的类型，数组是几维数组等。之后根据这些基本的类型构建出一些表达式的类型。这些表达式整数同样蕴含着这些信息。

之后通过定义一些简单的宏对这个类型系统进行简单的包装，增加一些判断是否为指针等类似的简单的操作函数。

最后封装了一层函数，将简单的宏叠加使用，完成更加复杂的判断和操作，诸如根据表达式的整数构造出新的表达式类型。

这样做的系统好处是，写起来非常迅速，很快就可以构建好整个系统，而且随着封装层数的增加，外层调用起来也不是非常麻烦。函数内部逻辑简单，很容易调试。

然而这样做的可维护性和可拓展性太差了，而且造成了很多无法逆转的问题。

六、实验总结:

此次实验中主要存在以下问题：

1、复用 C 语言的标准文法使得整个文法比较冗长，虽然已经存在 struct、union、enum 等的文法和语义动作但并不能合适地规约到整个文法的开始产生式；

2、类型系统的构建并不完善，可扩展性较差，没有维护好继承关系；

3、版本管理较混乱，应该使用 git 进行管理。

附录

```
%token END 0 "end of file"
%token <str> IDENTIFIER INT_CONST DOUBLE_CONST CHAR_CONST STR_CONST
%token <str> VOID CHAR INT FLOAT DOUBLE
%token <str> IF ELSE WHILE FOR
%token <str> CONTINUE BREAK RETURN
%token <str> OR_OP AND_OP EQ_OP NE_OP LE_OP GE_OP LEFT_OP RIGHT_OP INC_OP DEC_OP SIZEOF PTR_OP
%token <str> MUL_ASSIGN DIV_ASSIGN MOD_ASSIGN ADD_ASSIGN SUB_ASSIGN LEFT_ASSIGN RIGHT_ASSIGN AND_ASSIGN XOR_ASSIGN OR_ASSIGN
%token <str> LBRACE RBRACE LCURVE RCURVE LSQUARE RSQUARE COMMA SQM
%token <str> STAR ADD SUB DIV MOD ASSIGN QUESTION COLON
%token <str> INCLUSIVE_OR EXCLUSIVE_OR AND LESS GREATER DOT WAVE SIGH
%token <str> READ WRITE
%type <tree> program
%type <tree> external_declaration declaration parameter_declaration
%type <tree> function_definition
%type <tree> type_specifier
%type <tree> declarator direct_declarator init_declarator init_declarator_tail init_declarator_body
%type <tree> statement compound_statement expression_statement selection_statement iteration_statement jump_statement
%type <tree> read_statement write_statement
%type <tree> pointer
%type <tree> expression constant_expression conditional_expression assignment_expression unary_expression
%type <tree> logical_or_expression logical_and_expression inclusive_or_expression exclusive_or_expression
%type <tree> and_expression relational_expression shift_expression additive_expression equality_expression
%type <tree> multiplicative_expression cast_expression postfix_expression primary_expression
%type <tree> parameter_list statement_list declaration_list initializer_list
%type <tree> argument_expression_list
%type <tree> initializer
%type <tree> assignment_operator unary_operator

%locations

%%
```

```
code                    : END
                        | program END
                        ;

program                 : external_declaration
                        | program external_declaration
                        ;

external_declaration    : function_definition
                        | declaration
                        ;

function_definition     : type_specifier declarator compound_statement
                        ;

type_specifier          : VOID
                        | CHAR
                        | INT
                        | FLOAT
                        | DOUBLE
                        ;


declarator              : pointer direct_declarator
                        | LCURVE pointer direct_declarator RCURVE
                        | direct_declarator
                        | LCURVE direct_declarator RCURVE
                        ;

pointer                 : STAR
                        | pointer STAR
                        ;

direct_declarator       : IDENTIFIER
                        | direct_declarator LSQUARE RSQUARE

                        |  direct_declarator  LSQUARE  constant_expression  RSQUARE

                        | IDENTIFIER LCURVE parameter_list RCURVE
                        | IDENTIFIER LCURVE RCURVE
                        ;

constant_expression     : conditional_expression
                        ;

parameter_list          : parameter_declaration
                        | parameter_list COMMA parameter_declaration
                        ;

parameter_declaration : type_specifier  IDENTIFIER
                        ;

                        | LBRACE statement_list RBRACE
                        | LBRACE declaration_list statement_list RBRACE
                        ;

declaration_list        : declaration
                        | declaration_list declaration
```

```
                        ;

declaration             : init_declarator SQM
                        | init_declarator init_declarator_tail SQM
                        ;

init_declarator_tail    : COMMA init_declarator_body
                        | init_declarator_tail COMMA init_declarator_body
                        ;

init_declarator         : type_specifier init_declarator_body
                        ;

init_declarator_body    : declarator
                        | declarator ASSIGN initializer
                        ;

initializer             : assignment_expression
                        | LBRACE initializer_list RBRACE
                        | LBRACE initializer_list COMMA RBRACE

                        ;

initializer_list        : initializer
                        | initializer_list COMMA initializer
                        ;

statement_list          : statement
                        | statement_list statement
                        |
                        ;

statement               : compound_statement
                        | expression_statement
                        | selection_statement
                        | iteration_statement
                        | jump_statement
                        | read_statement
                        | write_statement
                        ;

 read_statement         : READ LCURVE IDENTIFIER RCURVE SQM

                        | READ LCURVE INT_CONST RCURVE SQM

                        | READ LCURVE DOUBLE_CONST RCURVE SQM

                        | READ LCURVE CHAR_CONST RCURVE SQM
                        | READ LCURVE STR_CONST RCURVE SQM
                        ;

write_statement         : WRITE LCURVE IDENTIFIER RCURVE SQM
                        | WRITE LCURVE INT_CONST RCURVE SQM
                        | WRITE LCURVE DOUBLE_CONST RCURVE SQM

                        | WRITE LCURVE CHAR_CONST RCURVE SQM

                        | WRITE LCURVE STR_CONST RCURVE SQM
```

```
                        ;

expression_statement  : SQM
                        | expression SQM
                        ;

selection_statement     : IF LCURVE expression RCURVE statement

                        | IF LCURVE expression RCURVE statement ELSE statement
                        ;

iteration_statement     : WHILE LCURVE expression RCURVE statement

                        | FOR LCURVE expression_statement expression_statement
RCURVE statement
                        | FOR LCURVE expression_statement expression_statement
expression RCURVE statement
                        ;

jump_statement          : CONTINUE SQM
                        | BREAK SQM
                        | RETURN SQM
                        | RETURN expression SQM
                        ;

expression              : assignment_expression
                        | expression COMMA assignment_expression
                        ;

assignment_expression : conditional_expression
                        | unary_expression assignment_operator assignment_expression

                        ;

conditional_expression : logical_or_expression
                        | logical_or_expression QUESTION expression COLON
conditional_expression

                        ;

logical_or_expression   : logical_and_expression
                        | logical_or_expression OR_OP logical_and_expression

                        ;

logical_and_expression  : inclusive_or_expression
                        | logical_and_expression AND_OP inclusive_or_expression

                        ;

inclusive_or_expression : exclusive_or_expression
                        | inclusive_or_expression INCLUSIVE_OR exclusive_or_expression

                        ;

exclusive_or_expression : and_expression
                        | exclusive_or_expression EXCLUSIVE_OR and_expression
                        ;
```

```
and_expression          : equality_expression
                        | and_expression AND equality_expression
                        ;

equality_expression     : relational_expression
                        | equality_expression EQ_OP relational_expression
                        | equality_expression NE_OP relational_expression
                        ;

relational_expression   : shift_expression
                        | relational_expression LESS shift_expression
                        | relational_expression GREATER shift_expression
                        | relational_expression LE_OP shift_expression
                        | relational_expression GE_OP shift_expression
                        ;

shift_expression        : additive_expression
                        | shift_expression LEFT_OP additive_expression
                        | shift_expression RIGHT_OP additive_expression
                        ;

additive_expression     : multiplicative_expression
                        | additive_expression ADD multiplicative_expression
                        | additive_expression SUB multiplicative_expression
                        ;

multiplicative_expression   : cast_expression
                            | multiplicative_expression STAR cast_expression
                            | multiplicative_expression DIV cast_expression
                            | multiplicative_expression MOD cast_expression
                            ;

cast_expression             : unary_expression
                            | LCURVE type_name RCURVE cast_expression
                            ;

unary_expression        : postfix_expression
                            | INC_OP unary_expression
                            | DEC_OP unary_expression
                            | unary_operator cast_expression
                            | SIZEOF unary_expression
                            | SIZEOF LCURVE type_name RCURVE
                            ;

postfix_expression      : primary_expression
                            | postfix_expression LSQUARE expression RSQUARE
                            | postfix_expression LCURVE RCURVE

                            |   postfix_expression   LCURVE   argument_expression_list
RCURVE
                            | postfix_expression DOT IDENTIFIER
                            | postfix_expression PTR_OP IDENTIFIER
                            | postfix_expression INC_OP
                            | postfix_expression DEC_OP
                            ;

primary_expression          : IDENTIFIER
```

```
                        | INT_CONST
                        | DOUBLE_CONST
                        | CHAR_CONST
                        | STR_CONST
                        | LCURVE expression RCURVE
                        ;

argument_expression_list    : assignment_expression
                        | argument_expression_list COMMA assignment_expression

                        ;

unary_operator              : AND
                        | STAR
                        | ADD
                        | SUB
                        | WAVE
                        | SIGH
                        ;

assignment_operator         : ASSIGN
                        | MUL_ASSIGN
                        | DIV_ASSIGN
                        | MOD_ASSIGN
                        | ADD_ASSIGN
                        | SUB_ASSIGN
                        | LEFT_ASSIGN
                        | RIGHT_ASSIGN
                        | AND_ASSIGN
                        | XOR_ASSIGN
                        | OR_ASSIGN
                        ;

storage_class_specifier     : TYPEDEF
                        | EXTERN
                        | STATIC
                        | AUTO
                        | REGISTER
                        ;

struct_or_union_specifier   : struct_or_union IDENTIFIER LBRACE struct_declaration_list
RBRACE
                        | struct_or_union LBRACE struct_declaration_list RBRACE
                        | struct_or_union IDENTIFIER
                        ;

struct_or_union             : STRUCT
                        | UNION
                        ;

struct_declaration_list     : struct_declaration
                        | struct_declaration_list struct_declaration
                        ;

struct_declaration          : specifier_qualifier_list struct_declarator_list SQM
                        ;

specifier_qualifier_list    : type_specifier specifier_qualifier_list
```

```
                        | type_specifier
                        | type_qualifier specifier_qualifier_list
                        | type_qualifier
                        ;

struct_declarator_list  : struct_declarator
                        | struct_declarator_list COMMA struct_declarator

                        ;

struct_declarator       : declarator
                        | COLON constant_expression
                        | declarator COLON constant_expression
                        ;

enum_specifier          : ENUM LBRACE enumerator_list RBRACE
                        | ENUM IDENTIFIER LBRACE enumerator_list RBRACE
                        | ENUM IDENTIFIER
                        ;

enumerator_list         : enumerator
                        | enumerator_list COMMA enumerator
                        ;

enumerator              : IDENTIFIER
                        | IDENTIFIER ASSIGN constant_expression
                        ;

type_qualifier          : CONST
                        | VOLATILE
                        ;

type_qualifier_list     : type_qualifier
                        | type_qualifier_list type_qualifier
                        ;

parameter_type_list     : parameter_list
                        | parameter_list COMMA ELLIPSIS
                        ;

parameter_list          : parameter_declaration
                        | parameter_list COMMA parameter_declaration
                        ;

type_name               : specifier_qualifier_list
                        | specifier_qualifier_list abstract_declarator
                        ;

abstract_declarator     : pointer
                        | direct_abstract_declarator
                        | pointer direct_abstract_declarator
                        ;

direct_abstract_declarator  : LCURVE abstract_declarator RCURVE
                        | LSQUARE RSQUARE
                        | LSQUARE constant_expression RSQUARE
                        | direct_abstract_declarator LSQUARE RSQUARE
                        | direct_abstract_declarator LSQUARE constant_expression
```

```
RSQUARE
                        | LCURVE RCURVE

                        | LCURVE parameter_type_list RCURVE
                        | direct_abstract_declarator LCURVE RCURVE
                        |  direct_abstract_declarator  LCURVE  parameter_type_list
RCURVE

                        ;

labeled_statement       : IDENTIFIER COLON statement
                        | CASE constant_expression COLON statement
                        | DEFAULT COLON statement
                        ;
%%
```