福州大学

《编译系统设计实践》

实验项目二：语法分析实验

学号：041701320

姓名：杨鑫杰

年级：2017 级

学院：数计学院

专业：软件工程

本组其它成员：学号 221701117 姓名 余嘉宸

学号 221701114 姓名 张玉麟

学号 221701131 姓名 郑志成

学号 221701121 姓名 沈明炜

实验时间：2019－2020 学年第二学期

任课教师：陈晖

目录

报告概要

本报告将具体描述本小组在编译原理实践课程中第三次实验《语法分析实验》的完成情况，介绍实验的目的，以及开发环境，还有具体的实验内容，包括实验过程中涉及的文法，实验的设计思路，还有具体的实验要求和实验代码分析过程和对应运行文件的运行结果的展示。

同时该报告还将介绍代码部分和文档部分等的团队分工情况，以及最后附录将提供学习过程中参考书目。

关键字：LR 分析、Goto、Action 表构造、闭包集

实验目的

根据给出的文法编制 LR（1）分析程序，以便对任意输入的符号串进行分析。本次实验的目的主要是加深对 LR（1）分析法的理解。

对已给语言文法，构造 LR(1)分析表，编制语法分析程序，要求将错误信息输出到语法错误文件中，并输出分析句子的过程（显示栈的内容）；

实验准备

本实验的运行环境为 dev-c++ ，采用 c++11 新标准，将文件分为多个头文件和一个主函数 cpp 。

准备的文法

```
program → block
block→{ decls  stmts}
decls → decls  decl  | ε
decl → type  id;
type → type[num]  |  basic
stmts → stmts  stmt | ε
stmt → loc=bool;
      | if(bool)stmt
      | if(bool)stmt else stmt
      | while(bool)stmt
      | do stmt while(bool);
      | break;
      | block
Loc → loc[num]  | id
bool →bool  ||  join  |  join
join → join  &&  equality  | equality
equality → equality==rel  | equality ! = rel  | rel
rel → expr<expr |expr<=expr|expr>=expr|expr>expr|expr
expr → expr+term |expr-term |term
term → term*unary|term/unary|unary
```

unary→！unary | -unary | factor
factor→ (bool) | loc | num | real | true |false

实验内容

1.1设计思路

通过实验 1 的学习和使用、用 DFA 扫描源程序，分离出单词放在符号表里。实验 1 将单词转化为文法可以识别的内容，既识别单词的属性。实验 2 首先对输入的文法进行分析。分析方法采用 LR 分析。然后再根据 LR 分析产生的 Goto 表和 Action 表对文法进行分析。分析的同时输出结果。

LR 分析首先对文法部分进行初始化构造 I0 集合，然后根据 I0 集合的内容求闭包。求闭包的时候会借助 First 函数求 First 集合来确保闭包的完整。字符栈每移进一个字符，就从二维数组分析表中锁定状态栈栈顶的状态（行）和当前进栈符号（列）对应的 action 或者 goto，如此反复，直到匹配或者出错。

1.2程序内容

命名空间：smw_project
作用：读取文法文件、进行预处理，主要是对文法的式子进行处理，预先存储所有的文法单词、符号、非终结符、产生式、项目集等，同时在产生式加·前，项目集加·后。
变量声明：

```
/*存储所有的符号*/
set<string> all_symbols;
/*存放所有的符号*/
vector<string> vec_symbols;
/*符号哈希*/
map<string, int> Hash;
/*存储终结符*/
set<string> terminal_symbol;
/*存储非终结符*/
```

set<string> nonterminal_symbol;
/*存放产生式，加·前 */
vector<Production> vec_production;
/*存储项目集，加·后 */
set<Production> set_production;
函数：
构造函数：void smw_project()：读取文法文件、识别并预处理文法文件。
命名空间：yxj_LRtable
变量声明：
int row, col;
//row 分析表的行数，col 分析表的列数
string LRtalbe[1005][1005];
//存储 LR(1)分析表的结果
vector<set<Project>> pset(1);
//项目的集合
函数声明
set<string> FirstSet(vector<string> X)
//求 First 集合
set<Project> yxj_Go(set<Project> SP, string S)
//Go 表的生成
set<Project> zyl_Closure(set<Project> I)
//函数功能：用于求 closure
void yxj_LR1()
//求出 LR1(1)分析表的过程
void Print_Pj()
//输出项目集、LR（1）分析表到文件中 yxj_result.txt
void Print_LR1()
//将 LR(1)分析表（与上述文件不同、上面的文件只是输出正确、下面的文件对错误的地方标出 Error）输出到文件 yxj_LR(1)Table.txt 中

1.3输入输出文件

输入文件：

文法（zzc_language.txt）

输出文件：

项目输出文件 yxj_result.txt

由于篇幅有限，只展示部分（前十项）

CLOSURE

****

```
0:
    block -> . { decls stmts }, #/{
    block -> . { decls }, #/{
    block -> . { stmts }, #/{
    block -> . { }, #/{
    decl -> . DOUBLE ID ;, DOUBLE/FLOAT/INT/{
    decl -> . FLOAT ID ;, DOUBLE/FLOAT/INT/{
    decl -> . INT ID ;, DOUBLE/FLOAT/INT/{
    decls -> . decl, {
    decls -> . decl decls, {
    function_definition -> . block, #/{
    function_definition_list -> . function_definition, #
    function_definition_list        ->        .        function_definition
function_definition_list, #
    program -> . decls function_definition_list, #
    program -> . function_definition_list, #
    program' -> . program, #
****************************************
1:
    decl -> DOUBLE . ID ;, DOUBLE/FLOAT/INT/{
****************************************
2:
    decl -> FLOAT . ID ;, DOUBLE/FLOAT/INT/{
****************************************
3:
    decl -> INT . ID ;, DOUBLE/FLOAT/INT/{
****************************************
4:
    function_definition -> block ., #/{
****************************************
5:
    decl -> . DOUBLE ID ;, DOUBLE/FLOAT/INT/{
    decl -> . FLOAT ID ;, DOUBLE/FLOAT/INT/{
    decl -> . INT ID ;, DOUBLE/FLOAT/INT/{
    decls -> . decl, {
    decls -> . decl decls, {
    decls -> decl ., {
    decls -> decl . decls, {
****************************************
6:
    block -> . { decls stmts }, #/{
    block -> . { decls }, #/{
    block -> . { stmts }, #/{
    block -> . { }, #/{
```

```
    function_definition -> . block, #/{
    function_definition_list -> . function_definition, #
    function_definition_list        ->        .        function_definition
function_definition_list, #
    program -> decls . function_definition_list, #
****************************************
7:
    block -> . { decls stmts }, #/{
    block -> . { decls }, #/{
    block -> . { stmts }, #/{
    block -> . { }, #/{
    function_definition -> . block, #/{
    function_definition_list -> . function_definition, #
    function_definition_list        ->        .        function_definition
function_definition_list, #
    function_definition_list -> function_definition ., #
    function_definition_list        ->        function_definition        .
function_definition_list, #
****************************************
8:
    program -> function_definition_list ., #
****************************************
9:
    program' -> program ., #
****************************************
10:
    assignment -> . ID = assignment, ,/;
    assignment -> . equality, ,/;
    block -> . { decls stmts }, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    block -> . { decls }, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    block -> . { stmts }, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    block -> . { }, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    block -> { . decls stmts }, #/{
    block -> { . decls }, #/{
    block -> { . stmts }, #/{
    block -> { . }, #/{
    bool -> . assignment, ;
    bool -> . assignment , bool, ;
    decl -> . DOUBLE ID ;, (/+/-/;/DOUBLE/FLOAT/ID/IF/INT/NUM/RETURN/WHILE/{/}
    decl -> . FLOAT ID ;, (/+/-/;/DOUBLE/FLOAT/ID/IF/INT/NUM/RETURN/WHILE/{/}
    decl -> . INT ID ;, (/+/-/;/DOUBLE/FLOAT/ID/IF/INT/NUM/RETURN/WHILE/{/}
    decls -> . decl, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    decls -> . decl decls, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    equality -> . rel, ,/;
```

```
    equality -> . rel != equality, ,/;
    equality -> . rel == equality, ,/;
    expr -> . term, !=/,/;/</<=/==/>/>=
    expr -> . term + expr, !=/,/;/</<=/==/>/>=
    expr -> . term - expr, !=/,/;/</<=/==/>/>=
    expression_statement -> . ;, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    expression_statement -> . bool ;, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    iteration -> . WHILE ( bool ) stmt, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    jump_statement -> . RETURN ;, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    jump_statement -> . RETURN bool ;, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    primary -> . ( expression ), !=/*/+/,/-///;/</<=/==/>/>=
    primary -> . ID, !=/*/+/,/-///;/</<=/==/>/>=
    primary -> . NUM, !=/*/+/,/-///;/</<=/==/>/>=
    rel -> . expr, !=/,/;/==
    rel -> . expr < rel, !=/,/;/==
    rel -> . expr <= rel, !=/,/;/==
    rel -> . expr > rel, !=/,/;/==
    rel -> . expr >= rel, !=/,/;/==
    selection -> . IF ( bool ) stmt, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    selection -> . IF ( bool ) stmt ELSE stmt,
(/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    stmt -> . block, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    stmt -> . expression_statement, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    stmt -> . iteration, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    stmt -> . jump_statement, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    stmt -> . selection, (/+/-/;/ID/IF/NUM/RETURN/WHILE/{/}
    stmts -> . stmt, }
    stmts -> . stmt stmts, }
    term -> . unary, !=/+/,/-/;/</<=/==/>/>=
    term -> . unary * term, !=/+/,/-/;/</<=/==/>/>=
    term -> . unary / term, !=/+/,/-/;/</<=/==/>/>=
    unary -> . primary, !=/*/+/,/-///;/</<=/==/>/>=
    unary -> . unary_operator unary, !=/*/+/,/-///;/</<=/==/>/>=
    unary_operator -> . +, (/+/-/ID/NUM
    unary_operator -> . -, (/+/-/ID/NUM
****************************************
```

输出文件：yxj_LR(1)Table.txt

第一行

//前面是 Action 表后面是 Goto 表

err err err err err err err err err err err err err err err err s1 err s2 err err s3 err err err err s10 err err 4 err 5 6 err err err err 7 8 err err err err err err 9 err err err err err err err err

测试

测试用例 1

```
void func(){
    int s; int i;
    i = 1; s = 1;
    while (i < 10) {
        s = s * i;
        i = i + 1;
        if(i > 1){i = 2;}
    }
}
```

测试用例结果：Acc

输出栈内容

```
        状态                                                                    符
号
输入串
1       0                                                                       #
VOID ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE ( ID < NUM ) { ID
= ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
2       0 2                                                                     #
VOID
ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
3       0 6                                                                     #
return_type
ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
4       0 6 12                                                                  #
return_type                                                                     ID
( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID *
ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
5       0 6 12 18                                                               #
return_type                                                                     ID
(
                                                                        ) { INT
ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID * ID ; ID =
ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
```

6 0 6 12 18 21 #
return_type ID ( )
{ INT ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ;
ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
7 0 6 13 #
return_type function_name
{ INT ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ;
ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
8 0 6 13 20 #
return_type function_name
{
INT ID ; INT ID ; ID
= NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID >
NUM) { ID = NUM ; } } } #
9 0 6 13 20 31 #
return_type function_name { INT
ID ; INT ID ; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID =
ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
10 0 6 13 20 31 72 #
return_type function_name { INT
ID
; INT ID ; ID = NUM ; ID =
NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID
= NUM ; } } } #
11 0 6 13 20 31 72 117 #
return_type function_name { INT ID ;
INT ID ; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID +
NUM ; IF (ID > NUM) { ID = NUM ; } } } #
12 0 6 13 20 51 #
return_type function_name { variable_definition
INT ID ; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID +
NUM ; IF (ID > NUM) { ID = NUM ; } } } #
13 0 6 13 20 51 31 #
return_type function_name { variable_definition INT
ID ; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ;
IF (ID > NUM) { ID = NUM ; } } } #
14 0 6 13 20 51 31 72 #
return_type function_name { variable_definition INT
ID
; ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID
= ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
15 0 6 13 20 51 31 72 117 #
return_type function_name { variable_definition INT ID ;
ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF

(ID > NUM) { ID = NUM ; } } } #
16 0 6 13 20 51 51 #
return_type function_name { variable_definition variable_definition ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
17 0 6 13 20 51 92 #
return_type function_name { variable_definition variable_definition_list ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
18 0 6 13 20 52 #
return_type function_name { variable_definition_list ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
19 0 6 13 20 52 29 #
return_type function_name { variable_definition_list ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
20 0 6 13 20 52 29 70 #
return_type function_name { variable_definition_list ID = NUM ; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
21 0 6 13 20 52 29 70 32 #
return_type function_name { variable_definition_list ID = NUM
; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
22 0 6 13 20 52 29 70 44 #
return_type function_name { variable_definition_list ID = primary_expression
; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
23 0 6 13 20 52 29 70 49 #
return_type function_name { variable_definition_list ID = unary_expression
; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
24 0 6 13 20 52 29 70 43 #
return_type function_name { variable_definition_list ID = multiplicative_expression
; ID = NUM ; WHILE (ID < NUM) { ID = ID * ID ; ID = ID + NUM ; IF (ID > NUM) { ID = NUM ; } } } #
25 0 6 13 20 52 29 70 35 #
return_type function_name { variable_definition_list ID = additive_expression

```
                                    ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
26      0 6 13 20 52 29 70 45                                                   #
return_type    function_name    {    variable_definition_list    ID    =
relational_expression
                                    ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
27      0 6 13 20 52 29 70 38                                                   #
return_type    function_name    {    variable_definition_list    ID    =
equality_expression
                                    ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
28      0 6 13 20 52 29 70 115                                                  #
return_type    function_name    {    variable_definition_list    ID    =
assignment_expression
                                    ; ID = NUM ; WHILE ( ID < NUM ) { ID = ID
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
29      0 6 13 20 52 36                                                         #
return_type        function_name        {        variable_definition_list
assignment_expression
                                    ; ID = NUM ; WHILE ( ID < NUM ) { ID
= ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
30      0 6 13 20 52 39                                                         #
return_type        function_name        {        variable_definition_list
expression
                                    ; ID = NUM ; WHILE ( ID < NUM ) { ID
= ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
31      0 6 13 20 52 39 81                                                      #
return_type   function_name   {   variable_definition_list   expression   ;
ID = NUM ; WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM )
{ ID = NUM ; } } } #
32      0 6 13 20 52 40                                                         #
return_type function_name { variable_definition_list expression_statement
ID = NUM ; WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM )
{ ID = NUM ; } } } #
33      0 6 13 20 52 47                                                         #
return_type    function_name    {    variable_definition_list    statement
ID = NUM ; WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM )
{ ID = NUM ; } } } #
34      0 6 13 20 52 47 29                                                      #
return_type   function_name   {   variable_definition_list   statement   ID
= NUM ; WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
35      0 6 13 20 52 47 29 70                                                   #
```

```
return_type function_name { variable_definition_list statement ID =
NUM ; WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
36      0 6 13 20 52 47 29 70 32                                        #
return_type function_name { variable_definition_list statement ID =
NUM
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
37      0 6 13 20 52 47 29 70 44                                        #
return_type function_name { variable_definition_list statement ID =
primary_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
38      0 6 13 20 52 47 29 70 49                                        #
return_type function_name { variable_definition_list statement ID =
unary_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
39      0 6 13 20 52 47 29 70 43                                        #
return_type function_name { variable_definition_list statement ID =
multiplicative_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
40      0 6 13 20 52 47 29 70 35                                        #
return_type function_name { variable_definition_list statement ID =
additive_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
41      0 6 13 20 52 47 29 70 45                                        #
return_type function_name { variable_definition_list statement ID =
relational_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
42      0 6 13 20 52 47 29 70 38                                        #
return_type function_name { variable_definition_list statement ID =
equality_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
43      0 6 13 20 52 47 29 70 115                                       #
return_type function_name { variable_definition_list statement ID =
assignment_expression
                                    ; WHILE ( ID < NUM ) { ID = ID * ID ; ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
44      0 6 13 20 52 47 36                                              #
```

```
return_type    function_name    {    variable_definition_list    statement
assignment_expression
                                        ; WHILE ( ID < NUM ) { ID = ID * ID ;
ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
45      0 6 13 20 52 47 39                                                  #
return_type    function_name    {    variable_definition_list    statement
expression
                                        ; WHILE ( ID < NUM ) { ID = ID * ID ;
ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
46      0 6 13 20 52 47 39 81                                               #
return_type function_name { variable_definition_list statement expression ;
WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } }
#
47      0 6 13 20 52 47 40                                                  #
return_type    function_name    {    variable_definition_list    statement
expression_statement
WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } }
#
48      0 6 13 20 52 47 47                                                  #
return_type  function_name  {  variable_definition_list  statement  statement
WHILE ( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } }
#
49      0 6 13 20 52 47 47 34                                               #
return_type function_name { variable_definition_list statement statement WHILE
( ID < NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } }
#
50      0 6 13 20 52 47 47 34 75                                            #
return_type function_name { variable_definition_list statement statement WHILE
(
                                    ID < NUM ) { ID = ID * ID ; ID = ID + NUM ;
IF ( ID > NUM ) { ID = NUM ; } } } #
51      0 6 13 20 52 47 47 34 75 59                                         #
return_type function_name { variable_definition_list statement statement WHILE
(                                                                       ID
< NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
52      0 6 13 20 52 47 47 34 75 66                                         #
return_type function_name { variable_definition_list statement statement WHILE
(                                                       primary_expression
< NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
53      0 6 13 20 52 47 47 34 75 68                                         #
return_type function_name { variable_definition_list statement statement WHILE
(                                                         unary_expression
< NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
54      0 6 13 20 52 47 47 34 75 65                                         #
```

```
return_type function_name { variable_definition_list statement statement WHILE
(                                                   multiplicative_expression
< NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
55      0 6 13 20 52 47 47 34 75 61                                          #
return_type function_name { variable_definition_list statement statement WHILE
(                                                     additive_expression
< NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
56      0 6 13 20 52 47 47 34 75 61 101                                      #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
NUM ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
57      0 6 13 20 52 47 47 34 75 61 101 60                                   #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
NUM
                    ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
58      0 6 13 20 52 47 47 34 75 61 101 66                                   #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
primary_expression
                    ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
59      0 6 13 20 52 47 47 34 75 61 101 68                                   #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
unary_expression
                    ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
60      0 6 13 20 52 47 47 34 75 61 101 65                                   #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
multiplicative_expression
                    ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
61      0 6 13 20 52 47 47 34 75 61 101 61                                   #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
additive_expression
                    ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
62      0 6 13 20 52 47 47 34 75 61 101 137                                  #
return_type function_name { variable_definition_list statement statement WHILE
(                           additive_expression                              <
```

```
relational_expression
                    ) { ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
63      0 6 13 20 52 47 47 34 75 67                                     #
return_type function_name { variable_definition_list statement statement WHILE
(                                                       relational_expression
                                                ) { ID = ID * ID ; ID = ID + NUM ;
IF ( ID > NUM ) { ID = NUM ; } } } #
64      0 6 13 20 52 47 47 34 75 63                                     #
return_type function_name { variable_definition_list statement statement WHILE
(                                                         equality_expression
                                                ) { ID = ID * ID ; ID = ID + NUM ;
IF ( ID > NUM ) { ID = NUM ; } } } #
65      0 6 13 20 52 47 47 34 75 62                                     #
return_type function_name { variable_definition_list statement statement WHILE
(                                                       assignment_expression
                                                ) { ID = ID * ID ; ID = ID + NUM ;
IF ( ID > NUM ) { ID = NUM ; } } } #
66      0 6 13 20 52 47 47 34 75 119                                    #
return_type function_name { variable_definition_list statement statement WHILE
(                                                               expression
                                                ) { ID = ID * ID ; ID = ID + NUM ;
IF ( ID > NUM ) { ID = NUM ; } } } #
67      0 6 13 20 52 47 47 34 75 119 149                                #
return_type function_name { variable_definition_list statement statement WHILE
(                                 expression                            )
{ ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
68      0 6 13 20 52 47 47 34 75 119 149 53                             #
return_type function_name { variable_definition_list statement statement WHILE
(                                 expression                            )
{
                                    ID = ID * ID ; ID = ID + NUM ; IF ( ID > NUM )
{ ID = NUM ; } } } #
69      0 6 13 20 52 47 47 34 75 119 149 53 29                          #
return_type function_name { variable_definition_list statement statement WHILE
(               expression               )               {               ID
= ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
70      0 6 13 20 52 47 47 34 75 119 149 53 29 70                       #
return_type function_name { variable_definition_list statement statement WHILE
(           expression           )           {           ID           =
ID * ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
71      0 6 13 20 52 47 47 34 75 119 149 53 29 70 29                    #
return_type function_name { variable_definition_list statement statement WHILE
(         expression         )         {         ID         =         ID
```

```
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
72      0 6 13 20 52 47 47 34 75 119 149 53 29 70 44                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression      )       {       ID      =       primary_expression
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
73      0 6 13 20 52 47 47 34 75 119 149 53 29 70 49                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression      )       {       ID      =       unary_expression
* ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
74      0 6 13 20 52 47 47 34 75 119 149 53 29 70 49 88                 #
return_type function_name { variable_definition_list statement statement WHILE
(      expression      )      {      ID      =      unary_expression      *
ID ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
75      0 6 13 20 52 47 47 34 75 119 149 53 29 70 49 88 90              #
return_type function_name { variable_definition_list statement statement WHILE
(      expression      )      {      ID      =      unary_expression      *
ID
                    ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
76      0 6 13 20 52 47 47 34 75 119 149 53 29 70 49 88 44              #
return_type function_name { variable_definition_list statement statement WHILE
(      expression      )      {      ID      =      unary_expression      *
primary_expression
                    ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
77      0 6 13 20 52 47 47 34 75 119 149 53 29 70 49 88 49              #
return_type function_name { variable_definition_list statement statement WHILE
(      expression      )      {      ID      =      unary_expression      *
unary_expression
                    ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
78      0 6 13 20 52 47 47 34 75 119 149 53 29 70 49 88 129             #
return_type function_name { variable_definition_list statement statement WHILE
(      expression      )      {      ID      =      unary_expression      *
multiplicative_expression
                    ; ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
79      0 6 13 20 52 47 47 34 75 119 149 53 29 70 43                    #
return_type function_name { variable_definition_list statement statement WHILE
(           expression          )          {          ID          =
multiplicative_expression
                                    ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
80      0 6 13 20 52 47 47 34 75 119 149 53 29 70 35                    #
return_type function_name { variable_definition_list statement statement WHILE
(           expression          )          {          ID          =
additive_expression
                                    ; ID = ID + NUM ; IF ( ID > NUM ) { ID
```

```
= NUM ; } } } #
81      0 6 13 20 52 47 47 34 75 119 149 53 29 70 45                    #
return_type function_name { variable_definition_list statement statement WHILE
(           expression           )           {           ID           =
relational_expression
                                        ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
82      0 6 13 20 52 47 47 34 75 119 149 53 29 70 38                    #
return_type function_name { variable_definition_list statement statement WHILE
(           expression           )           {           ID           =
equality_expression
                                        ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
83      0 6 13 20 52 47 47 34 75 119 149 53 29 70 115                   #
return_type function_name { variable_definition_list statement statement WHILE
(           expression           )           {           ID           =
assignment_expression
                                        ; ID = ID + NUM ; IF ( ID > NUM ) { ID
= NUM ; } } } #
84      0 6 13 20 52 47 47 34 75 119 149 53 36                          #
return_type function_name { variable_definition_list statement statement WHILE
(                               expression                              )
{                                                   assignment_expression
                                                ; ID = ID + NUM ; IF ( ID > NUM )
{ ID = NUM ; } } } #
85      0 6 13 20 52 47 47 34 75 119 149 53 39                          #
return_type function_name { variable_definition_list statement statement WHILE
(                               expression                              )
{                                                             expression
                                                ; ID = ID + NUM ; IF ( ID > NUM )
{ ID = NUM ; } } } #
86      0 6 13 20 52 47 47 34 75 119 149 53 39 81                       #
return_type function_name { variable_definition_list statement statement WHILE
(          expression          )          {          expression          ;
ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
87      0 6 13 20 52 47 47 34 75 119 149 53 40                          #
return_type function_name { variable_definition_list statement statement WHILE
(          expression          )          {          expression_statement
ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
88      0 6 13 20 52 47 47 34 75 119 149 53 47                          #
return_type function_name { variable_definition_list statement statement WHILE
(             expression             )             {             statement
ID = ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
89      0 6 13 20 52 47 47 34 75 119 149 53 47 29                       #
```

```
return_type function_name { variable_definition_list statement statement WHILE
(          expression          )          {          statement          ID
= ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
90      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70                    #
return_type function_name { variable_definition_list statement statement WHILE
(        expression        )        {        statement        ID        =
ID + NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
91      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 29                 #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       ID       =       ID
+ NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
92      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 44                 #
return_type function_name { variable_definition_list statement statement WHILE
(     expression     )     {     statement     ID     =     primary_expression
+ NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
93      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 49                 #
return_type function_name { variable_definition_list statement statement WHILE
(     expression     )     {     statement     ID     =     unary_expression
+ NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
94      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43                 #
return_type function_name { variable_definition_list statement statement WHILE
(    expression    )    {    statement    ID    =    multiplicative_expression
+ NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
95      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43 82              #
return_type function_name { variable_definition_list statement statement WHILE
(   expression   )   {   statement   ID   =   multiplicative_expression   +
NUM ; IF ( ID > NUM ) { ID = NUM ; } } } #
96      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43 82 32           #
return_type function_name { variable_definition_list statement statement WHILE
(   expression   )   {   statement   ID   =   multiplicative_expression   +
NUM
                ; IF ( ID > NUM ) { ID = NUM ; } } } #
97      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43 82 44           #
return_type function_name { variable_definition_list statement statement WHILE
(   expression   )   {   statement   ID   =   multiplicative_expression   +
primary_expression
                ; IF ( ID > NUM ) { ID = NUM ; } } } #
98      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43 82 49           #
return_type function_name { variable_definition_list statement statement WHILE
(   expression   )   {   statement   ID   =   multiplicative_expression   +
unary_expression
                ; IF ( ID > NUM ) { ID = NUM ; } } } #
99      0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43 82 43           #
return_type function_name { variable_definition_list statement statement WHILE
```

```
( expression ) { statement ID = multiplicative_expression + multiplicative_expression
                        ; IF ( ID > NUM ) { ID = NUM ; } } } #
100     0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 43 82 125          #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement ID = multiplicative_expression + additive_expression
                  ; IF ( ID > NUM ) { ID = NUM ; } } } #
101     0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 35                 #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement ID = additive_expression
                                                ; IF ( ID > NUM ) { ID = NUM ; } } }
#
102     0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 45                 #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement ID = relational_expression
                                                ; IF ( ID > NUM ) { ID = NUM ; } } }
#
103     0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 38                 #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement ID = equality_expression
                                                ; IF ( ID > NUM ) { ID = NUM ; } } }
#
104     0 6 13 20 52 47 47 34 75 119 149 53 47 29 70 115                #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement ID = assignment_expression
                                                ; IF ( ID > NUM ) { ID = NUM ; } } }
#
105     0 6 13 20 52 47 47 34 75 119 149 53 47 36                       #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement assignment_expression
                                                    ; IF ( ID > NUM ) { ID = NUM ; } } } #
106     0 6 13 20 52 47 47 34 75 119 149 53 47 39                       #
return_type function_name { variable_definition_list statement statement WHILE ( expression ) { statement expression
                                                    ; IF ( ID > NUM ) { ID = NUM ; } } } #
```

```
107     0 6 13 20 52 47 47 34 75 119 149 53 47 39 81                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       expression       ;
IF ( ID > NUM ) { ID = NUM ; } } } #
108     0 6 13 20 52 47 47 34 75 119 149 53 47 40                       #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       expression_statement
IF ( ID > NUM ) { ID = NUM ; } } } #
109     0 6 13 20 52 47 47 34 75 119 149 53 47 47                       #
return_type function_name { variable_definition_list statement statement WHILE
(         expression         )         {         statement         statement
IF ( ID > NUM ) { ID = NUM ; } } } #
110     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
( ID > NUM ) { ID = NUM ; } } } #
111     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71                 #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(
                                            ID > NUM ) { ID = NUM ; } } } #
112     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 59              #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(                                                                   ID
                                          > NUM ) { ID = NUM ; } } } #
113     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 66              #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(                                                   primary_expression
                                          > NUM ) { ID = NUM ; } } } #
114     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 68              #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(                                                     unary_expression
                                          > NUM ) { ID = NUM ; } } } #
115     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 65              #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(                                               multiplicative_expression
                                          > NUM ) { ID = NUM ; } } } #
116     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61              #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
```

```
(                                                    additive_expression
                                                  > NUM ) { ID = NUM ; } } } #
117     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103          #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
NUM ) { ID = NUM ; } } } #
118     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103 60       #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
NUM
                              ) { ID = NUM ; } } } #
119     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103 66       #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
primary_expression
                                ) { ID = NUM ; } } } #
120     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103 68       #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
unary_expression
                              ) { ID = NUM ; } } } #
121     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103 65       #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
multiplicative_expression
                                      ) { ID = NUM ; } } } #
122     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103 61       #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
additive_expression
                                ) { ID = NUM ; } } } #
123     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 61 103 139      #
return_type function_name ( variable_definition_list statement statement WHILE
( expression ) { statement statement IF ( additive_expression >
relational_expression
                                  ) { ID = NUM ; } } } #
124     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 67              #
return_type function_name ( variable_definition_list statement statement WHILE
(       expression      )       {       statement       statement       IF
(                                                   relational_expression
                                                    ) { ID = NUM ; } } } #
125     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 63              #
return_type function_name ( variable_definition_list statement statement WHILE
(       expression      )       {       statement       statement       IF
```

```
(                                                    equality_expression
                                                  ) { ID = NUM ; } } } #
126     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 62              #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(                                                  assignment_expression
                                                  ) { ID = NUM ; } } } #
127     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116             #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement       IF
(                                                         expression
                                                  ) { ID = NUM ; } } } #
128     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148         #
return_type function_name { variable_definition_list statement statement WHILE
(   expression   )   {   statement   statement   IF   (   expression   )
{ ID = NUM ; } } } #
129     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162     #
return_type function_name { variable_definition_list statement statement WHILE
(   expression   )   {   statement   statement   IF   (   expression   )
{
                                            ID = NUM ; } } } #
130     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29  # return_type
function_name  {  variable_definition_list  statement  statement  WHILE
(  expression  )  {  statement  statement  IF  (  expression  )  {  ID
= NUM ; } } } #
131     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 #
return_type function_name { variable_definition_list statement statement WHILE
(  expression  )  {  statement  statement  IF  (  expression  )  {  ID  =
NUM ; } } } #
132     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 32
# return_type function_name { variable_definition_list statement statement
WHILE  (  expression  )  {  statement  statement  IF  (  expression  )  {  ID  =
NUM
                                                ; } } } #
133     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 44
# return_type function_name { variable_definition_list statement statement
WHILE  (  expression  )  {  statement  statement  IF  (  expression  )  {  ID  =
primary_expression
                                                 ; } } } #
134     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 49
# return_type function_name { variable_definition_list statement statement
WHILE  (  expression  )  {  statement  statement  IF  (  expression  )  {  ID  =
unary_expression
                                                ; } } } #
```

```
135     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 43
# return_type function_name { variable_definition_list statement statement
WHILE ( expression ) { statement statement IF ( expression ) { ID =
multiplicative_expression
                                                    ; } } } #
136     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 35
# return_type function_name { variable_definition_list statement statement
WHILE ( expression ) { statement statement IF ( expression ) { ID =
additive_expression
                                                ; } } } #
137     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 45
# return_type function_name { variable_definition_list statement statement
WHILE ( expression ) { statement statement IF ( expression ) { ID =
relational_expression
                                                  ; } } } #
138     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 38
# return_type function_name { variable_definition_list statement statement
WHILE ( expression ) { statement statement IF ( expression ) { ID =
equality_expression
                                                ; } } } #
139     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 29 70 115
# return_type function_name { variable_definition_list statement statement
WHILE ( expression ) { statement statement IF ( expression ) { ID =
assignment_expression
                                                  ; } } } #
140     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 36  # return_type
function_name  {  variable_definition_list  statement  statement  WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                                                   assignment_expression
                                                  ; } } } #
141     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 39  # return_type
function_name  {  variable_definition_list  statement  statement  WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                                                           expression
                                                  ; } } } #
142     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 39 81 #
return_type function_name { variable_definition_list statement statement WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                                 expression                                ;
                                                    } } } #
143     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 40  # return_type
function_name  {  variable_definition_list  statement  statement  WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                                                     expression_statement
```

```
                                                    } } } #
144     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 47  # return_type
function_name  {  variable_definition_list  statement  statement  WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                                                                   statement
                                                    } } } #
145     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 170 # return_type
function_name  {  variable_definition_list  statement  statement  WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                                                       statement_list
                                                    } } } #
146     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 162 170 177 #
return_type function_name { variable_definition_list statement statement WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
{                           statement_list                          }
                                                      } } #
147     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 155     #
return_type function_name { variable_definition_list statement statement WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
compound_statement
                                                    } } #
148     0 6 13 20 52 47 47 34 75 119 149 53 47 47 30 71 116 148 161     #
return_type function_name { variable_definition_list statement statement WHILE
(  expression  )  {  statement  statement  IF  (  expression  )
statement
                                                    } } #
149     0 6 13 20 52 47 47 34 75 119 149 53 47 47 46                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement
selection_statement
                                                                    }
 } #
150     0 6 13 20 52 47 47 34 75 119 149 53 47 47 47                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement
statement
                                                                    }
 } #
151     0 6 13 20 52 47 47 34 75 119 149 53 47 47 86                    #
return_type function_name { variable_definition_list statement statement WHILE
(       expression       )       {       statement       statement
statement_list
                                                                    }
 } #
```

```
152     0 6 13 20 52 47 47 34 75 119 149 53 47 86                          #
return_type function_name { variable_definition_list statement statement WHILE
(             expression             )             {             statement
statement_list

        } } #
153     0 6 13 20 52 47 47 34 75 119 149 53 95                             #
return_type function_name { variable_definition_list statement statement WHILE
(                                expression                                )
{                                                                statement_list

                    } } #
154     0 6 13 20 52 47 47 34 75 119 149 53 95 132                         #
return_type function_name { variable_definition_list statement statement WHILE
(                                expression                                )
{                              statement_list                              }

                      } #
155     0 6 13 20 52 47 47 34 75 119 149 37                                #
return_type function_name { variable_definition_list statement statement WHILE
(                                expression                                )
compound_statement

                      } #
156     0 6 13 20 52 47 47 34 75 119 149 163                               #
return_type function_name { variable_definition_list statement statement WHILE
(                                expression                                )
statement

                      } #
157     0 6 13 20 52 47 47 41                                              #
return_type  function_name  {  variable_definition_list  statement  statement
iteration_statement

                                          } #
158     0 6 13 20 52 47 47 47                                              #
return_type  function_name  {  variable_definition_list  statement  statement
statement

                                          } #
159     0 6 13 20 52 47 47 86                                              #
return_type  function_name  {  variable_definition_list  statement  statement
statement_list
```

```
                                                    } #
160     0 6 13 20 52 47 86                                                  #
return_type    function_name    {    variable_definition_list    statement
statement_list

                                                          } #
161     0 6 13 20 52 93                                                     #
return_type        function_name        {        variable_definition_list
statement_list

                                                                  } #
162     0 6 13 20 52 93 131                                                 #
return_type  function_name  {  variable_definition_list  statement_list  }
#
163     0 6 13 19                                                           #
return_type                function_name                compound_statement
#
164     0 3                                                                 #
function_definition
#
165     0 4                                                                 #
function_definition_list
#
166     0 5                                                                 #
program
```

测试用例 2

测试用例 2

```
void func(){
    int s; int i;
    i = 1; s = 1;
    if( i == 1 ){
       s = 2;
    }
}
```

//测试结果:Acc

输出栈内容

```
1       0                                                                   #
VOID ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID =
NUM ; } } #
2       0 2                                                                 #
VOID
ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } }
```

```
#
3       0 6                                                                 #
return_type
ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } }
#
4       0 6 12                                                              #
return_type                                                                 ID
( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } }
#
5       0 6 12 18                                                           #
return_type                                                                 ID
(

                          ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID
== NUM ) { ID = NUM ; } } #
6       0 6 12 18 21                                                        #
return_type                    ID                    (                    )
{ INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
7       0 6 13                                                              #
return_type                                                 function_name
{ INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
8       0 6 13 20                                                           #
return_type                                                 function_name
{

INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
9       0 6 13 20 31                                                        #
return_type               function_name               {               INT
ID ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
10      0 6 13 20 31 72                                                     #
return_type               function_name               {               INT
ID

                    ; INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID =
NUM ; } } #
11      0 6 13 20 31 72 117                                                 #
return_type         function_name         {         INT         ID        ;
INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
12      0 6 13 20 51                                                        #
return_type           function_name           {           variable_definition
INT ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
13      0 6 13 20 51 31                                                     #
return_type       function_name       {       variable_definition       INT
ID ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
```

```
14      0 6 13 20 51 31 72                                                  #
return_type       function_name       {       variable_definition       INT
ID

         ; ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
15      0 6 13 20 51 31 72 117                                              #
return_type    function_name    {    variable_definition    INT    ID    ;
ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
16      0 6 13 20 51 51                                                     #
return_type   function_name   {   variable_definition   variable_definition
ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
17      0 6 13 20 51 92                                                     #
return_type  function_name  {  variable_definition  variable_definition_list
ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
18      0 6 13 20 52                                                        #
return_type         function_name         {         variable_definition_list
ID = NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
19      0 6 13 20 52 29                                                     #
return_type      function_name      {      variable_definition_list      ID
= NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
20      0 6 13 20 52 29 70                                                  #
return_type    function_name    {    variable_definition_list    ID    =
NUM ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
21      0 6 13 20 52 29 70 32                                               #
return_type    function_name    {    variable_definition_list    ID    =
NUM

              ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
22      0 6 13 20 52 29 70 44                                               #
return_type    function_name    {    variable_definition_list    ID    =
primary_expression

              ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
23      0 6 13 20 52 29 70 49                                               #
return_type    function_name    {    variable_definition_list    ID    =
unary_expression

              ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
24      0 6 13 20 52 29 70 43                                               #
return_type    function_name    {    variable_definition_list    ID    =
multiplicative_expression

              ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
25      0 6 13 20 52 29 70 35                                               #
```

```
return_type    function_name    {    variable_definition_list    ID    =
additive_expression

             ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
26      0 6 13 20 52 29 70 45                                          #
return_type    function_name    {    variable_definition_list    ID    =
relational_expression

             ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
27      0 6 13 20 52 29 70 38                                          #
return_type    function_name    {    variable_definition_list    ID    =
equality_expression

             ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
28      0 6 13 20 52 29 70 115                                         #
return_type    function_name    {    variable_definition_list    ID    =
assignment_expression

             ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
29      0 6 13 20 52 36                                                #
return_type        function_name        {        variable_definition_list
assignment_expression

                 ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
30      0 6 13 20 52 39                                                #
return_type        function_name        {        variable_definition_list
expression

                 ; ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
31      0 6 13 20 52 39 81                                             #
return_type   function_name   {   variable_definition_list   expression   ;
ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
32      0 6 13 20 52 40                                                #
return_type function_name { variable_definition_list expression_statement
ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
33      0 6 13 20 52 47                                                #
return_type    function_name    {    variable_definition_list    statement
ID = NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
34      0 6 13 20 52 47 29                                             #
return_type   function_name   {   variable_definition_list   statement   ID
= NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
35      0 6 13 20 52 47 29 70                                          #
return_type  function_name  {  variable_definition_list  statement  ID  =
NUM ; IF ( ID == NUM ) { ID = NUM ; } } #
```

```
36      0 6 13 20 52 47 29 70 32                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
NUM

              ; IF ( ID == NUM ) { ID = NUM ; } } #
37      0 6 13 20 52 47 29 70 44                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
primary_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
38      0 6 13 20 52 47 29 70 49                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
unary_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
39      0 6 13 20 52 47 29 70 43                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
multiplicative_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
40      0 6 13 20 52 47 29 70 35                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
additive_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
41      0 6 13 20 52 47 29 70 45                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
relational_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
42      0 6 13 20 52 47 29 70 38                                        #
return_type  function_name  {  variable_definition_list  statement  ID  =
equality_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
43      0 6 13 20 52 47 29 70 115                                       #
return_type  function_name  {  variable_definition_list  statement  ID  =
assignment_expression

              ; IF ( ID == NUM ) { ID = NUM ; } } #
44      0 6 13 20 52 47 36                                              #
return_type    function_name    {    variable_definition_list    statement
assignment_expression
```

```
                    ; IF ( ID == NUM ) { ID = NUM ; } } #
45      0 6 13 20 52 47 39                                                  #
return_type    function_name    {    variable_definition_list    statement
expression

                    ; IF ( ID == NUM ) { ID = NUM ; } } #
46      0 6 13 20 52 47 39 81                                               #
return_type function_name { variable_definition_list statement expression ;
IF ( ID == NUM ) { ID = NUM ; } } #
47      0 6 13 20 52 47 40                                                  #
return_type    function_name    {    variable_definition_list    statement
expression_statement
IF ( ID == NUM ) { ID = NUM ; } } #
48      0 6 13 20 52 47 47                                                  #
return_type  function_name  {  variable_definition_list  statement  statement
IF ( ID == NUM ) { ID = NUM ; } } #
49      0 6 13 20 52 47 47 30                                               #
return_type function_name { variable_definition_list statement statement IF
( ID == NUM ) { ID = NUM ; } } #
50      0 6 13 20 52 47 47 30 71                                            #
return_type function_name { variable_definition_list statement statement IF
(

ID == NUM ) { ID = NUM ; } } #
51      0 6 13 20 52 47 47 30 71 59                                         #
return_type function_name { variable_definition_list statement statement IF
(                                                                       ID
== NUM ) { ID = NUM ; } } #
52      0 6 13 20 52 47 47 30 71 66                                         #
return_type function_name { variable_definition_list statement statement IF
(                                                       primary_expression
== NUM ) { ID = NUM ; } } #
53      0 6 13 20 52 47 47 30 71 68                                         #
return_type function_name { variable_definition_list statement statement IF
(                                                         unary_expression
== NUM ) { ID = NUM ; } } #
54      0 6 13 20 52 47 47 30 71 65                                         #
return_type function_name { variable_definition_list statement statement IF
(                                                multiplicative_expression
== NUM ) { ID = NUM ; } } #
55      0 6 13 20 52 47 47 30 71 61                                         #
return_type function_name { variable_definition_list statement statement IF
(                                                       additive_expression
== NUM ) { ID = NUM ; } } #
```

```
56      0 6 13 20 52 47 47 30 71 67                                    #
return_type function_name { variable_definition_list statement statement IF
(                                                    relational_expression
== NUM ) { ID = NUM ; } } #
57      0 6 13 20 52 47 47 30 71 67 110                                #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
NUM ) { ID = NUM ; } } #
58      0 6 13 20 52 47 47 30 71 67 110 60                             #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
NUM
                                                                      )
{ ID = NUM ; } } #
59      0 6 13 20 52 47 47 30 71 67 110 66                             #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
primary_expression
                                                                      )
{ ID = NUM ; } } #
60      0 6 13 20 52 47 47 30 71 67 110 68                             #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
unary_expression
                                                                      )
{ ID = NUM ; } } #
61      0 6 13 20 52 47 47 30 71 67 110 65                             #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
multiplicative_expression
                                                                      )
{ ID = NUM ; } } #
62      0 6 13 20 52 47 47 30 71 67 110 61                             #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
additive_expression
                                                                      )
{ ID = NUM ; } } #
63      0 6 13 20 52 47 47 30 71 67 110 67                             #
return_type function_name { variable_definition_list statement statement IF
(                        relational_expression                        ==
relational_expression
                                                                      )
{ ID = NUM ; } } #
```

```
64      0 6 13 20 52 47 47 30 71 67 110 145                                  #
return_type function_name { variable_definition_list statement statement IF
(                         relational_expression                           ==
equality_expression
                                                                          )
{ ID = NUM ; } } #
65      0 6 13 20 52 47 47 30 71 63                                          #
return_type function_name { variable_definition_list statement statement IF
(                                                    equality_expression

                        ) { ID = NUM ; } } #
66      0 6 13 20 52 47 47 30 71 62                                          #
return_type function_name { variable_definition_list statement statement IF
(                                                  assignment_expression

                        ) { ID = NUM ; } } #
67      0 6 13 20 52 47 47 30 71 116                                         #
return_type function_name { variable_definition_list statement statement IF
(                                                              expression

                        ) { ID = NUM ; } } #
68      0 6 13 20 52 47 47 30 71 116 148                                     #
return_type function_name { variable_definition_list statement statement IF
(                               expression                                  )
{ ID = NUM ; } } #
69      0 6 13 20 52 47 47 30 71 116 148 162                                 #
return_type function_name { variable_definition_list statement statement IF
(                               expression                                  )
{

ID = NUM ; } } #
70      0 6 13 20 52 47 47 30 71 116 148 162 29                              #
return_type function_name { variable_definition_list statement statement IF
(               expression              )               {               ID
= NUM ; } } #
71      0 6 13 20 52 47 47 30 71 116 148 162 29 70                           #
return_type function_name { variable_definition_list statement statement IF
(           expression           )           {           ID           =
NUM ; } } #
72      0 6 13 20 52 47 47 30 71 116 148 162 29 70 32                        #
return_type function_name { variable_definition_list statement statement IF
(           expression           )           {           ID           =
NUM
```

```
                ; } } #
73      0 6 13 20 52 47 47 30 71 116 148 162 29 70 44                   #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
primary_expression

                ; } } #
74      0 6 13 20 52 47 47 30 71 116 148 162 29 70 49                   #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
unary_expression

                ; } } #
75      0 6 13 20 52 47 47 30 71 116 148 162 29 70 43                   #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
multiplicative_expression

                ; } } #
76      0 6 13 20 52 47 47 30 71 116 148 162 29 70 35                   #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
additive_expression

                ; } } #
77      0 6 13 20 52 47 47 30 71 116 148 162 29 70 45                   #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
relational_expression

                ; } } #
78      0 6 13 20 52 47 47 30 71 116 148 162 29 70 38                   #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
equality_expression

                ; } } #
79      0 6 13 20 52 47 47 30 71 116 148 162 29 70 115                  #
return_type function_name { variable_definition_list statement statement IF
(           expression          )           {           ID          =
assignment_expression

                ; } } #
80      0 6 13 20 52 47 47 30 71 116 148 162 36                         #
```

```
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                                                    assignment_expression

                     ; } } #
81      0 6 13 20 52 47 47 30 71 116 148 162 39                          #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                                                                expression

                     ; } } #
82      0 6 13 20 52 47 47 30 71 116 148 162 39 81                       #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                              expression                                ;

                       } } #
83      0 6 13 20 52 47 47 30 71 116 148 162 40                          #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                                                      expression_statement

                       } } #
84      0 6 13 20 52 47 47 30 71 116 148 162 47                          #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                                                                 statement

                       } } #
85      0 6 13 20 52 47 47 30 71 116 148 162 170                         #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                                                           statement_list

                       } } #
86      0 6 13 20 52 47 47 30 71 116 148 162 170 177                     #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
{                            statement_list                              }

                         } #
87      0 6 13 20 52 47 47 30 71 116 148 155                             #
return_type function_name { variable_definition_list statement statement IF
(                              expression                                )
```

```
compound_statement

                        } #
88      0 6 13 20 52 47 47 30 71 116 148 161                                    #
return_type function_name { variable_definition_list statement statement IF
(                               expression                                      )
statement

                        } #
89      0 6 13 20 52 47 47 46                                                   #
return_type  function_name  {  variable_definition_list  statement  statement
selection_statement

                                        } #
90      0 6 13 20 52 47 47 47                                                   #
return_type  function_name  {  variable_definition_list  statement  statement
statement

                                        } #
91      0 6 13 20 52 47 47 86                                                   #
return_type  function_name  {  variable_definition_list  statement  statement
statement_list

                                        } #
92      0 6 13 20 52 47 86                                                      #
return_type    function_name    {    variable_definition_list    statement
statement_list

                                                } #
93      0 6 13 20 52 93                                                         #
return_type         function_name         {         variable_definition_list
statement_list

                                                                } #
94      0 6 13 20 52 93 131                                                     #
return_type  function_name  {  variable_definition_list  statement_list  }
#
95      0 6 13 19                                                               #
return_type                 function_name                 compound_statement
#
96      0 3                                                                     #
function_definition
#
97      0 4                                                                     #
```

```
function_definition_list
#
98      0 5                                                                    #
program
```

测试用例 3

```
void func(){
int s; int i;
i = 1; s = 1;
if( i == 1 ){
s = 2;
}else
}
```

测试结果：错误

action 值：err

error 语法分析结束且失败!语句在处理# { decls stmt stmt IF (bool) { bool ; 语句时出现了错误!

输出栈：

状态	符号	输入串
1	0	# VOID ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
2	0 2	# VOID ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
3	0 6	# return_type ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
4	0 6 12	# return_type ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
5	0 6 12 18	# return_type ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
6	0 6 12 18 21	# return_type ID ( ) { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
7	0 6 13	# return_type function_name { INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #

8 0 6 13 20 #
return_type function_name
{

INT ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
9 0 6 13 20 31 #
return_type function_name { INT
ID ; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
10 0 6 13 20 31 72 #
return_type function_name { INT
ID

; INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
11 0 6 13 20 31 72 117 #
return_type function_name { INT ID ;
INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
12 0 6 13 20 51 #
return_type function_name { variable_definition
INT ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
13 0 6 13 20 51 31 #
return_type function_name { variable_definition INT
ID ; ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
14 0 6 13 20 51 31 72 #
return_type function_name { variable_definition INT
ID

;
ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
15 0 6 13 20 51 31 72 117 #
return_type function_name { variable_definition INT ID ;
ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
16 0 6 13 20 51 51 #
return_type function_name { variable_definition variable_definition
ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
17 0 6 13 20 51 92 #
return_type function_name { variable_definition variable_definition_list
ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
18 0 6 13 20 52 #
return_type function_name { variable_definition_list
ID = NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
19 0 6 13 20 52 29 #
return_type function_name { variable_definition_list ID
= NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
20 0 6 13 20 52 29 70 #
return_type function_name { variable_definition_list ID =

NUM ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
21 0 6 13 20 52 29 70 32 #
return_type function_name { variable_definition_list ID = NUM

; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
22 0 6 13 20 52 29 70 44 #
return_type function_name { variable_definition_list ID = primary_expression

;
ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
23 0 6 13 20 52 29 70 49 #
return_type function_name { variable_definition_list ID = unary_expression

; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
24 0 6 13 20 52 29 70 43 #
return_type function_name { variable_definition_list ID = multiplicative_expression

;
ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
25 0 6 13 20 52 29 70 35 #
return_type function_name { variable_definition_list ID = additive_expression

;
ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
26 0 6 13 20 52 29 70 45 #
return_type function_name { variable_definition_list ID = relational_expression

;
ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
27 0 6 13 20 52 29 70 38 #
return_type function_name { variable_definition_list ID = equality_expression

;
ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
28 0 6 13 20 52 29 70 115 #
return_type function_name { variable_definition_list ID = assignment_expression

;
ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
29 0 6 13 20 52 36 #
return_type function_name { variable_definition_list assignment_expression

; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
30 0 6 13 20 52 39 #
return_type function_name { variable_definition_list expression

; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
31 0 6 13 20 52 39 81 #
return_type function_name { variable_definition_list expression ; ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
32 0 6 13 20 52 40 #
return_type function_name { variable_definition_list expression_statement ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
33 0 6 13 20 52 47 #
return_type function_name { variable_definition_list statement ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
34 0 6 13 20 52 47 29 #
return_type function_name { variable_definition_list statement ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
35 0 6 13 20 52 47 29 70 #
return_type function_name { variable_definition_list statement ID = NUM ; IF (ID == NUM) { ID = NUM ; } ELSE } #
36 0 6 13 20 52 47 29 70 32 # return_type function_name { variable_definition_list statement ID = NUM

; IF (ID == NUM) { ID = NUM ; } ELSE } #
37 0 6 13 20 52 47 29 70 44 # return_type function_name { variable_definition_list statement ID = primary_expression

; IF (ID == NUM) { ID = NUM ; } ELSE } #
38 0 6 13 20 52 47 29 70 49 # return_type function_name { variable_definition_list statement ID = unary_expression

; IF (ID == NUM) { ID = NUM ; } ELSE } #
39 0 6 13 20 52 47 29 70 43 # return_type function_name { variable_definition_list statement ID = multiplicative_expression
;
IF (ID == NUM) { ID = NUM ; } ELSE } #
40 0 6 13 20 52 47 29 70 35 # return_type function_name { variable_definition_list statement ID =

additive_expression

;

IF (ID == NUM) { ID = NUM ; } ELSE } #

41 0 6 13 20 52 47 29 70 45 # return_type function_name { variable_definition_list statement ID = relational_expression

;

IF (ID == NUM) { ID = NUM ; } ELSE } #

42 0 6 13 20 52 47 29 70 38 # return_type function_name { variable_definition_list statement ID = equality_expression

;

IF (ID == NUM) { ID = NUM ; } ELSE } #

43 0 6 13 20 52 47 29 70 115 # return_type function_name { variable_definition_list statement ID = assignment_expression

; IF (ID == NUM) { ID = NUM ; } ELSE } #

44 0 6 13 20 52 47 36 # return_type function_name { variable_definition_list statement assignment_expression

; IF (ID == NUM) { ID = NUM ; } ELSE } #

45 0 6 13 20 52 47 39 # return_type function_name { variable_definition_list statement expression

; IF (ID == NUM) { ID = NUM ; } ELSE } #

46 0 6 13 20 52 47 39 81 # return_type function_name { variable_definition_list statement expression ; IF (ID == NUM) { ID = NUM ; } ELSE } #

47 0 6 13 20 52 47 40 # return_type function_name { variable_definition_list statement expression_statement IF (ID == NUM) { ID = NUM ; } ELSE } #

48 0 6 13 20 52 47 47 # return_type function_name { variable_definition_list statement statement IF (ID == NUM) { ID = NUM ; } ELSE } #

49 0 6 13 20 52 47 47 30 # return_type function_name { variable_definition_list statement statement IF (ID == NUM) { ID = NUM ; } ELSE } #

50 0 6 13 20 52 47 47 30 71 # return_type function_name { variable_definition_list statement statement IF (

ID == NUM) { ID = NUM ; } ELSE } #
51 0 6 13 20 52 47 47 30 71 59 # return_type function_name { variable_definition_list statement statement IF (ID == NUM) { ID = NUM ; } ELSE } #
52 0 6 13 20 52 47 47 30 71 66 # return_type function_name { variable_definition_list statement statement IF (primary_expression == NUM) { ID = NUM ; } ELSE } #
53 0 6 13 20 52 47 47 30 71 68 # return_type function_name { variable_definition_list statement statement IF (unary_expression == NUM) { ID = NUM ; } ELSE } #
54 0 6 13 20 52 47 47 30 71 65 # return_type function_name { variable_definition_list statement statement IF (multiplicative_expression == NUM) { ID = NUM ; } ELSE } #
55 0 6 13 20 52 47 47 30 71 61 # return_type function_name { variable_definition_list statement statement IF (additive_expression == NUM) { ID = NUM ; } ELSE } #
56 0 6 13 20 52 47 47 30 71 67 # return_type function_name { variable_definition_list statement statement IF (relational_expression == NUM) { ID = NUM ; } ELSE } #
57 0 6 13 20 52 47 47 30 71 67 110 # return_type function_name { variable_definition_list statement statement IF (relational_expression == NUM) { ID = NUM ; } ELSE } #
58 0 6 13 20 52 47 47 30 71 67 110 60 # return_type function_name { variable_definition_list statement statement IF (relational_expression == NUM
) { ID = NUM ; }
ELSE } #
59 0 6 13 20 52 47 47 30 71 67 110 66 # return_type function_name { variable_definition_list statement statement IF (relational_expression == primary_expression
) { ID = NUM ; }
ELSE } #
60 0 6 13 20 52 47 47 30 71 67 110 68 # return_type function_name { variable_definition_list statement statement IF (relational_expression == unary_expression
) { ID = NUM ; }
ELSE } #
61 0 6 13 20 52 47 47 30 71 67 110 65 # return_type function_name { variable_definition_list statement statement IF (relational_expression == multiplicative_expression
) { ID = NUM ; }
ELSE } #
62 0 6 13 20 52 47 47 30 71 67 110 61 # return_type function_name { variable_definition_list statement statement IF (relational_expression ==

additive_expression

) { ID = NUM ; }

ELSE } #

63 0 6 13 20 52 47 47 30 71 67 110 67 # return_type function_name { variable_definition_list statement statement IF (relational_expression == relational_expression

) { ID = NUM ; }

ELSE } #

64 0 6 13 20 52 47 47 30 71 67 110 145 # return_type function_name { variable_definition_list statement statement IF (relational_expression == equality_expression

) { ID = NUM ; }

ELSE } #

65 0 6 13 20 52 47 47 30 71 63 # return_type function_name { variable_definition_list statement statement IF (equality_expression

) { ID = NUM ; } ELSE } #

66 0 6 13 20 52 47 47 30 71 62 # return_type function_name { variable_definition_list statement statement IF (assignment_expression

) { ID = NUM ; } ELSE } #

67 0 6 13 20 52 47 47 30 71 116 # return_type function_name { variable_definition_list statement statement IF (expression

) { ID = NUM ; } ELSE } #

68 0 6 13 20 52 47 47 30 71 116 148 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = NUM ; } ELSE } #

69 0 6 13 20 52 47 47 30 71 116 148 162 # return_type function_name { variable_definition_list statement statement IF (expression) {

ID = NUM ; } ELSE } #

70 0 6 13 20 52 47 47 30 71 116 148 162 29 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = NUM ; } ELSE } #

71 0 6 13 20 52 47 47 30 71 116 148 162 29 70 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = NUM ; } ELSE } #

72 0 6 13 20 52 47 47 30 71 116 148 162 29 70 32 # return_type function_name { variable_definition_list statement statement IF (expression) { ID =

NUM

; } ELSE } #
73 0 6 13 20 52 47 47 30 71 116 148 162 29 70 44 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = primary_expression

; } ELSE } #
74 0 6 13 20 52 47 47 30 71 116 148 162 29 70 49 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = unary_expression

; } ELSE } #
75 0 6 13 20 52 47 47 30 71 116 148 162 29 70 43 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = multiplicative_expression
; }
ELSE } #
76 0 6 13 20 52 47 47 30 71 116 148 162 29 70 35 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = additive_expression
;
} ELSE } #
77 0 6 13 20 52 47 47 30 71 116 148 162 29 70 45 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = relational_expression
;
} ELSE } #
78 0 6 13 20 52 47 47 30 71 116 148 162 29 70 38 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = equality_expression
;
} ELSE } #
79 0 6 13 20 52 47 47 30 71 116 148 162 29 70 115 # return_type function_name { variable_definition_list statement statement IF (expression) { ID = assignment_expression

; } ELSE } #
80 0 6 13 20 52 47 47 30 71 116 148 162 36 # return_type function_name { variable_definition_list statement statement IF (expression) { assignment_expression

; } ELSE } #
81 0 6 13 20 52 47 47 30 71 116 148 162 39 # return_type

function_name { variable_definition_list statement statement IF (expression) { expression

; } ELSE } #
82 0 6 13 20 52 47 47 30 71 116 148 162 39 81 # return_type
function_name { variable_definition_list statement statement IF (expression) { expression ;

} ELSE } #
83 0 6 13 20 52 47 47 30 71 116 148 162 40 # return_type
function_name { variable_definition_list statement statement IF (expression) { expression_statement

} ELSE } #
84 0 6 13 20 52 47 47 30 71 116 148 162 47 # return_type
function_name { variable_definition_list statement statement IF (expression) { statement

} ELSE } #
85 0 6 13 20 52 47 47 30 71 116 148 162 170 # return_type
function_name { variable_definition_list statement statement IF (expression) { statement_list

} ELSE } #
86 0 6 13 20 52 47 47 30 71 116 148 162 170 177 # return_type
function_name { variable_definition_list statement statement IF (expression) { statement_list } ELSE } #
87 0 6 13 20 52 47 47 30 71 116 148 155 # return_type
function_name { variable_definition_list statement statement IF (expression) compound_statement
ELSE } #
88 0 6 13 20 52 47 47 30 71 116 148 161 # return_type
function_name { variable_definition_list statement statement IF (expression) statement ELSE } #
89 0 6 13 20 52 47 47 30 71 116 148 161 169 # return_type
function_name { variable_definition_list statement statement IF (expression) statement ELSE

} #

团队分工

学号	姓名	分工
041701320	杨鑫杰	First 函数、Aciton 函数编写
221701117	余嘉宸	设计、文法分析思路构造
221701131	郑志成	LR 分析函数编写
221701114	张玉麟	测试、Go 函数编写
221701121	沈明炜	输入输出处理、主函数编写

实验总结

这次实验做的时间最长、遇到问题也比实验一多，群里组织了很多次的会议、通过不断的总结和学习最终能够较好的完成实验也是不容易。主要是课程的知识需要加强学习、理论转实践是有一定的区别的。

然后团队合作也是遇到一些小小的问题、对于分工之后的任务分配有的成员可能不够明确自己应该做什么、尽早的提出才不会因为需求不明确而在错误的方向上进行努力。积极完成各自的分工。

附录

8.1　参考书目

1.Alfred V.Aho 等著，赵建华译《编译原理》，机械工业出版社，2009 年

2.Andrew W.Appe1 著，赵克佳等译《现代编译原理 C 语言描述》人民邮电出版社，2006 年