Team Report

1. Functions

In this project, we implemented a top-down LL(1) method for analyzing Ada programs. The Ada program is divided into six statements at the top layer: assignment statement, if statement, while statement, procedure statement, until statement and for statement. After inputting tokens, the statement is recognized based on their first sets. Then, the program utilizes follow sets to identify the corresponding block. Besides, our program can look ahead one next input token to improve the parsing efficiency. At last, the output document is then segmented according to our defined rules with appropriate tree structure. The rules are enumerated as follows.

2. Rules

2.1 Lexical rule

Number	Type	Number	Type	Number	Type
1	if	14	-	27	integer
2	Else	15	*	28	string
3	then	16	/	29	:=
4	do	17	=	30	:
5	While	18	/=	31	,
6	loop	19	<	32	(
7	for	20	>	33	)
8	until	21	<=	34	;
9	begin	22	>=	35	EOF
10	end	23	IDENTIFIER	36	Is
11	procedure	24	NUMBER	37	ERROR
12	Call	25	STRING	38	xor
13	+	26	Float	39	range

2.2 Semantic rules

command sequence → <command> | <command sequence>; <command>

command → <assignment command> | <conditional command> | <iteration command> | <function command> | <repeat command> | <loop command>

<assignment command> → variable := <calculation> | variable := textConstant

conditional command → if <test> then <command sequence> end if | if <test> then <command sequence> else <command sequence> end if

iteration command → while <test> iterate <command sequence> end iterate

function command → execute function_name (<parameters>)

repeat command → repeat <command sequence> until <test>

loop command → for (<assignment command>; <test>; <assignment command>) execute <command sequence>

parameters → variable | <parameters>, variable

test → variable <comparison operator> variable | variable <comparison operator> numberConstant | variable <comparison operator> textConstant

comparison operator → > | >= | == | != | < | <=

calculation → <element> | <calculation> + <element> | <calculation> - <element>

element → <component> | <element> * <component> | <element> / <component>

component → variable | numberConstant | (<calculation>)

3. Results

3.1 Results of right test program:

RDPSTART

RDPFILE test_right.txt

RDPBEGIN StatementPart

    RDPTOKEN begin

```
RDPBEGIN StatementList
    RDPBEGIN Statement
        RDPBEGIN ProcedureStatement
            RDPTOKEN call
            RDPTOKEN IDENTIFIER 'get'
            RDPTOKEN (
            RDPBEGIN ArgumentList
                RDPTOKEN IDENTIFIER 'x1'
            RDPEND ArgumentList
            RDPTOKEN )
        RDPEND ProcedureStatement
    RDPEND Statement
    RDPTOKEN ;
    RDPBEGIN StatementList
        RDPBEGIN Statement
            RDPBEGIN AssignmentStatement
                RDPTOKEN IDENTIFIER 'x2'
                RDPTOKEN :=
                RDPBEGIN Expression
                    RDPBEGIN Term
                        RDPBEGIN Factor
                            RDPTOKEN NUMBER '1'
                        RDPEND Factor
                    RDPEND Term
                RDPEND Expression
            RDPEND AssignmentStatement
        RDPEND Statement
        RDPTOKEN ;
        RDPBEGIN StatementList
            RDPBEGIN Statement
```

```
        RDPBEGIN AssignmentStatement
            RDPTOKEN IDENTIFIER 'x3'
            RDPTOKEN :=
            RDPBEGIN Expression
                RDPBEGIN Term
                    RDPBEGIN Factor
                        RDPTOKEN NUMBER '0'
                    RDPEND Factor
                RDPEND Term
            RDPEND Expression
        RDPEND AssignmentStatement
    RDPEND Statement
    RDPTOKEN ;
    RDPBEGIN StatementList
        RDPBEGIN Statement
            RDPBEGIN WhileStatement
                RDPTOKEN while
                RDPBEGIN Condition
                    RDPTOKEN IDENTIFIER 'x1'
                    RDPBEGIN ConditionalOperator
                        RDPTOKEN /=
                    RDPEND ConditionalOperator
                    RDPTOKEN NUMBER '0'
                RDPEND Condition
                RDPTOKEN loop
                RDPBEGIN StatementList
                    RDPBEGIN Statement
                        RDPBEGIN AssignmentStatement
                            RDPTOKEN IDENTIFIER 'x2'
                            RDPTOKEN :=
```

```
                                        RDPBEGIN Expression
                                            RDPBEGIN Term
                                                RDPBEGIN Factor
                                                    RDPTOKEN    IDENTIFIER
'x2'
                                                RDPEND Factor
                                                RDPTOKEN *
                                                RDPBEGIN Term
                                                    RDPBEGIN Factor
                                                        RDPTOKEN
IDENTIFIER 'x1'
                                                    RDPEND Factor
                                                RDPEND Term
                                            RDPEND Term
                                        RDPEND Expression
                                    RDPEND AssignmentStatement
                                RDPEND Statement
                                RDPTOKEN ;
                                RDPBEGIN StatementList
                                    RDPBEGIN Statement
                                        RDPBEGIN AssignmentStatement
                                            RDPTOKEN IDENTIFIER 'x1'
                                            RDPTOKEN :=
                                            RDPBEGIN Expression
                                                RDPBEGIN Term
                                                    RDPBEGIN Factor
                                                        RDPTOKEN
IDENTIFIER 'x1'
                                                    RDPEND Factor
                                                RDPEND Term
```

```
                                                RDPTOKEN -
                                                RDPBEGIN Expression
                                                    RDPBEGIN Term
                                                        RDPBEGIN Factor
                                                            RDPTOKEN
NUMBER '1'
                                                        RDPEND Factor
                                                    RDPEND Term
                                                RDPEND Expression
                                            RDPEND Expression
                                        RDPEND AssignmentStatement
                                    RDPEND Statement
                                RDPEND StatementList
                            RDPEND StatementList
                            RDPTOKEN end
                            RDPTOKEN loop
                        RDPEND WhileStatement
                    RDPEND Statement
                    RDPTOKEN ;
                    RDPBEGIN StatementList
                        RDPBEGIN Statement
                            RDPBEGIN ForStatement
                                RDPTOKEN for
                                RDPTOKEN (
                                RDPBEGIN AssignmentStatement
                                    RDPTOKEN IDENTIFIER 'x2'
                                    RDPTOKEN :=
                                    RDPBEGIN Expression
                                        RDPBEGIN Term
                                            RDPBEGIN Factor
```

```
                                RDPTOKEN NUMBER '1'
                            RDPEND Factor
                        RDPEND Term
                    RDPEND Expression
                RDPEND AssignmentStatement
                RDPTOKEN ;
                RDPBEGIN Condition
                    RDPTOKEN IDENTIFIER 'x2'
                    RDPBEGIN ConditionalOperator
                        RDPTOKEN <
                    RDPEND ConditionalOperator
                    RDPTOKEN IDENTIFIER 'x1'
                RDPEND Condition
                RDPTOKEN ;
                RDPBEGIN AssignmentStatement
                    RDPTOKEN IDENTIFIER 'x2'
                    RDPTOKEN :=
                    RDPBEGIN Expression
                        RDPBEGIN Term
                            RDPBEGIN Factor
                                RDPTOKEN IDENTIFIER 'x2'
                            RDPEND Factor
                        RDPEND Term
                        RDPTOKEN +
                        RDPBEGIN Expression
                            RDPBEGIN Term
                                RDPBEGIN Factor
                                    RDPTOKEN NUMBER '1'
                                RDPEND Factor
                            RDPEND Term
```

```
                    RDPEND Expression
                RDPEND Expression
            RDPEND AssignmentStatement
            RDPTOKEN )
            RDPTOKEN do
            RDPBEGIN StatementList
                RDPBEGIN Statement
                    RDPBEGIN ProcedureStatement
                        RDPTOKEN call
                        RDPTOKEN IDENTIFIER 'put'
                        RDPTOKEN (
                        RDPBEGIN ArgumentList
                            RDPTOKEN IDENTIFIER 'x1'
                            RDPTOKEN ,
                            RDPBEGIN ArgumentList
                                RDPTOKEN    IDENTIFIER
'x2'
                                RDPTOKEN ,
                                RDPBEGIN ArgumentList
                                    RDPTOKEN
IDENTIFIER 'x3'
                                RDPEND ArgumentList
                            RDPEND ArgumentList
                        RDPEND ArgumentList
                        RDPTOKEN )
                    RDPEND ProcedureStatement
                RDPEND Statement
                RDPTOKEN ;
                RDPBEGIN StatementList
                    RDPBEGIN Statement
```

RDPBEGIN AssignmentStatement

RDPTOKEN IDENTIFIER 'x3'

RDPTOKEN :=

RDPBEGIN Expression

RDPBEGIN Term

RDPBEGIN Factor

RDPTOKEN

IDENTIFIER 'x3'

RDPEND Factor

RDPEND Term

RDPTOKEN +

RDPBEGIN Expression

RDPBEGIN Term

RDPBEGIN Factor

RDPTOKEN

IDENTIFIER 'x2'

RDPEND Factor

RDPEND Term

RDPEND Expression

RDPEND Expression

RDPEND AssignmentStatement

RDPEND Statement

RDPEND StatementList

RDPEND StatementList

RDPTOKEN end

RDPTOKEN loop

RDPEND ForStatement

RDPEND Statement

RDPTOKEN ;

RDPBEGIN StatementList

```
                RDPBEGIN Statement
                    RDPBEGIN ProcedureStatement
                        RDPTOKEN call
                        RDPTOKEN IDENTIFIER 'put'
                        RDPTOKEN (
                        RDPBEGIN ArgumentList
                            RDPTOKEN IDENTIFIER 'x3'
                        RDPEND ArgumentList
                        RDPTOKEN )
                    RDPEND ProcedureStatement
                RDPEND Statement
            RDPEND StatementList
        RDPEND StatementList
    RDPEND StatementList
RDPEND StatementList
RDPEND StatementList
RDPTOKEN end
RDPEND StatementPart
RDPTOKEN EOF
RDPSUCCESS

RDPFINISH
```

3.2 Results of wrong test program (test_wrong_3.txt)

```
RDPSTART
RDPFILE test_wrong_3.txt
RDPBEGIN StatementPart
    RDPTOKEN begin
```

```
RDPBEGIN StatementList
    RDPBEGIN Statement
        RDPBEGIN ProcedureStatement
            RDPTOKEN call
            RDPTOKEN IDENTIFIER 'get'
            RDPTOKEN (
            RDPBEGIN ArgumentList
                RDPTOKEN IDENTIFIER 'x1'
            RDPEND ArgumentList
            RDPTOKEN )
        RDPEND ProcedureStatement
    RDPEND Statement
    RDPTOKEN ;
    RDPBEGIN StatementList
        RDPBEGIN Statement
            RDPBEGIN AssignmentStatement
                RDPTOKEN IDENTIFIER 'x2'
                RDPTOKEN :=
                RDPBEGIN Expression
                    RDPBEGIN Term
                        RDPBEGIN Factor
                            RDPTOKEN NUMBER '1'
                        RDPEND Factor
                    RDPEND Term
                RDPEND Expression
            RDPEND AssignmentStatement
        RDPEND Statement
        RDPTOKEN ;
        RDPBEGIN StatementList
            RDPBEGIN Statement
```

```
RDPBEGIN WhileStatement
    RDPTOKEN while
    RDPBEGIN Condition
        RDPTOKEN IDENTIFIER 'x1'
        RDPBEGIN ConditionalOperator
            RDPTOKEN /=
        RDPEND ConditionalOperator
        RDPTOKEN NUMBER '0'
    RDPEND Condition
    RDPTOKEN loop
    RDPBEGIN StatementList
        RDPBEGIN Statement
            RDPBEGIN AssignmentStatement
                RDPTOKEN IDENTIFIER 'x1'
                RDPTOKEN :=
                RDPBEGIN Expression
                    RDPBEGIN Term
                        RDPBEGIN Factor
                            RDPTOKEN IDENTIFIER 'x1'
                        RDPEND Factor
                    RDPEND Term
                    RDPTOKEN -
                    RDPBEGIN Expression
                        RDPBEGIN Term
                            RDPBEGIN Factor
                                RDPTOKEN NUMBER '1'
                            RDPEND Factor
                        RDPEND Term
                    RDPEND Expression
                RDPEND Expression
```

RDPEND AssignmentStatement

RDPEND Statement

RDPTOKEN ;

RDPBEGIN StatementList

RDPBEGIN Statement

Compilation Exception

Caused by In test_wrong_3.txt: - Parsing error : statement part

Caused by In test_wrong_3.txt: - Parsing error : statement list

Caused by In test_wrong_3.txt: - Parsing error : statement list

Caused by In test_wrong_3.txt: - Parsing error : statement list

Caused by In test_wrong_3.txt: - Parsing error : statement

Caused by In test_wrong_3.txt: - Parsing error : while statement

Caused by In test_wrong_3.txt: - Parsing error : statement list

Caused by In test_wrong_3.txt: - Parsing error : statement list

Caused by In test_wrong_3.txt: - Parsing error : statement

Caused by In test_wrong_3.txt: - Expected token(s): ' if ' , ' assignment ' , ' until ', ' while ' or ' procedure ' but found: (' end ').

RDPFINISH