CS 419 Compiler

Project Form

Project Idea:
Project 3 - Language

Team Members NO#: 21

ID	Name	Level& Department	Section(Day-from-to)	Role (Lead/Member)	Grade
2019 00327	زياد مشهور حسن علي	Level 3 - CS	Wed. 12 - 2	Leader	
2019 00257	حامد محمود احمد عبدالهادي	Level 3 - CS	Wed. 12 - 2	Member	
2019 00218	باهر تامر هاشم مصطفى	Level 3 - CS	Wed. 12 - 2	Member	
2019 00214	باسل احمد عبدالعزيز	Level 3 - CS	Wed. 12 - 2	Member	
2019 00344	سلمي احمد عيسي	Level 3 - CS	Wed. 12 - 2	Member	
2019 00784	مرام محمد ابراهيم	Level 3 - CS	Wed. 4 - 6	Member	
2019 00962	ياسمين محي الدين	Level 3 - CS	-	Member	

Regular Expressions

- digit=: [0-9]*
- .digit = .[0-9]*
- letter = [_|(A-Z)|(a-z)]
- identifiers =: letter[letter | constants]*
- class =: Type
- Inheritance =: Infer
- Condition =: If
- Condition =: Else
- Integer =: Ipok
- SInteger =: Sipok
- Character =: Craf
- String =: Sequence
- Float =: Ipokf
- SFloat =: Sipokf
- Void =: Valueless
- Boolean =: Rational
- Break =: Endthis
- Loop =: However
- Loop =: When
- Return =: Respondwith
- Struct =: Srap
- Switch =: Scan
- Switch =: Conditionof
- Inclusion =: Require
- Stat Symbol =: @| ^

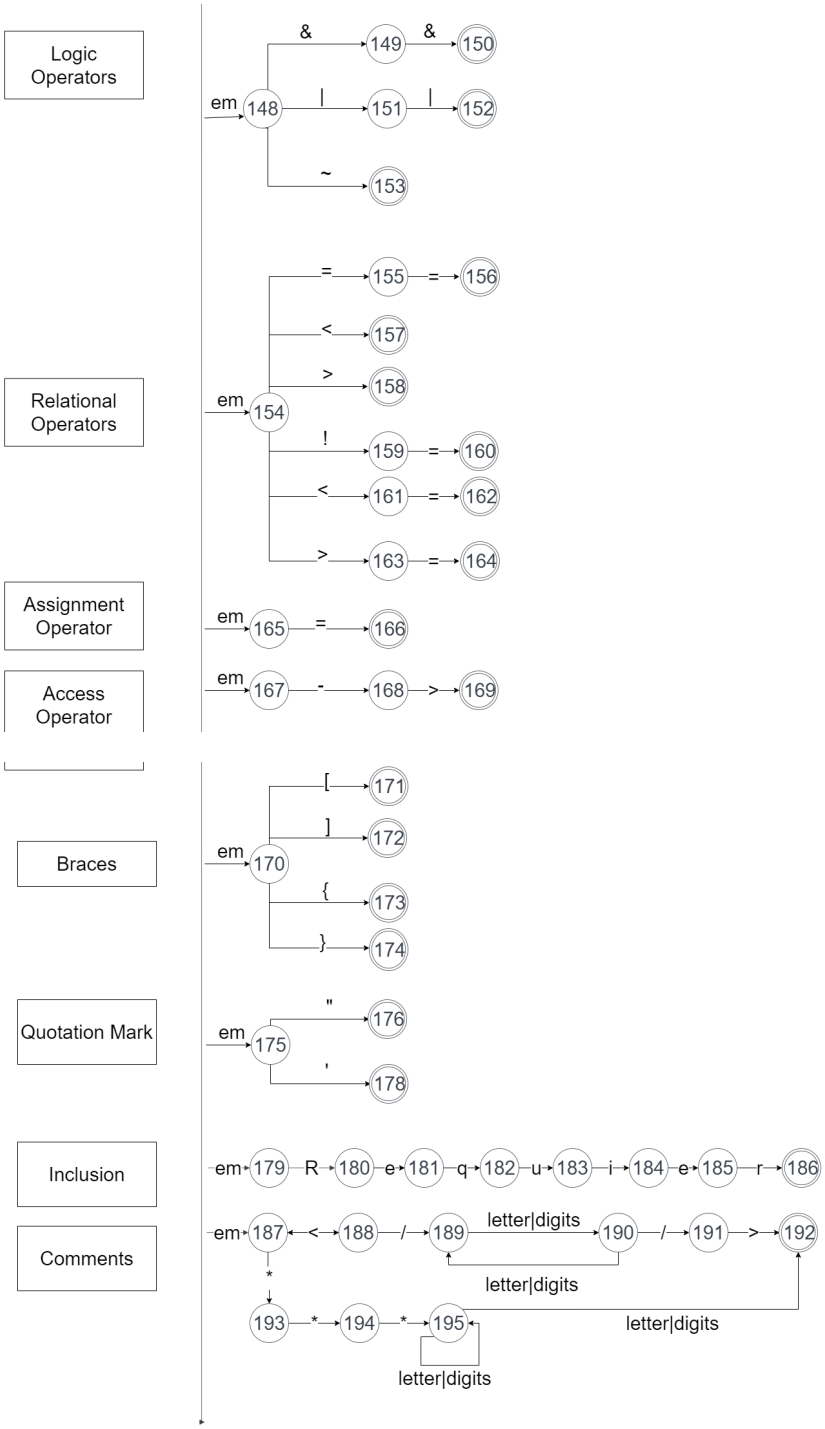
- End Symbol =: $|#
- Arithmetic Operation =: +|-||/
- Logic operators =: &&| || | ~
- relational operators =: == | < | > | != | <= | >=
- Assignment operator =: =
- Access Operator =: ->
- Braces =: { | } [|]
- Quotation Mark =: " | '
- Comment =: </ (letters|digits)/> | **(letters|digits)*
- token= [letter | constants| identifiers| class| inheritance|condition| integer| sinteger| Character| string| Float| SFloat| void| Boolean| Break| Loop| Return| Struct| Switch| Inclusion| Start Symbol| End Symbol| Arithmetic operation| Logic Operators| relational operators| Assignment operator| Access operator| Braces| Quotation Marl| Comment|]

NFA

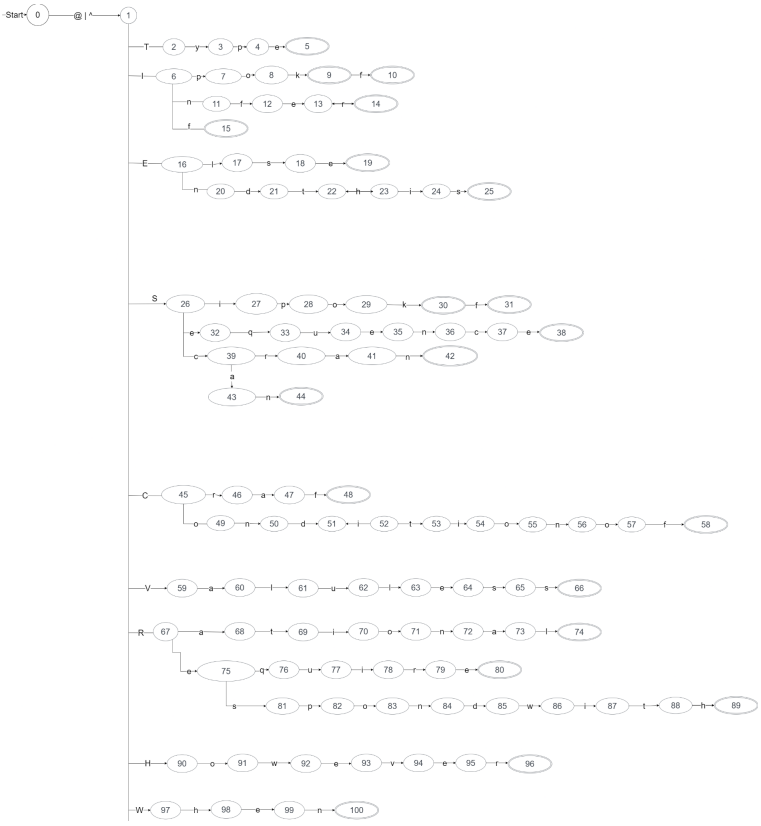
identifiers
digit =: [0-9]*
letter = [_|(A-Z)|(a-z)]
identifiers =: letter[letter | digit]*

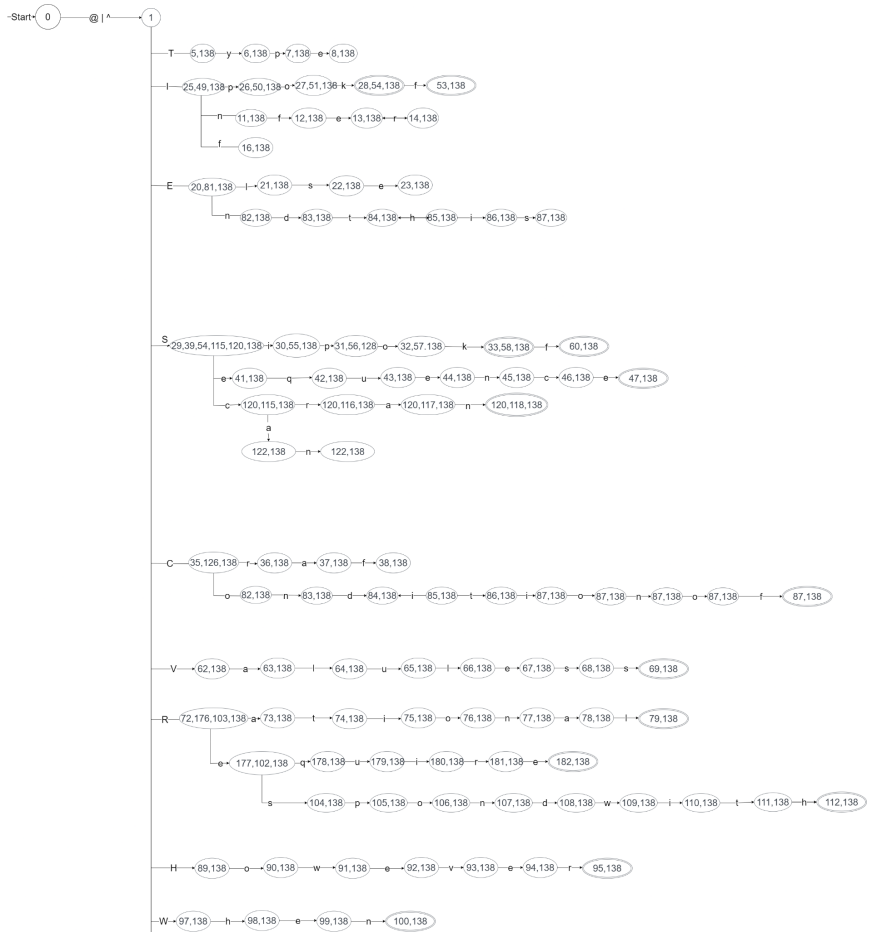
.digit

Arithmatic Operators

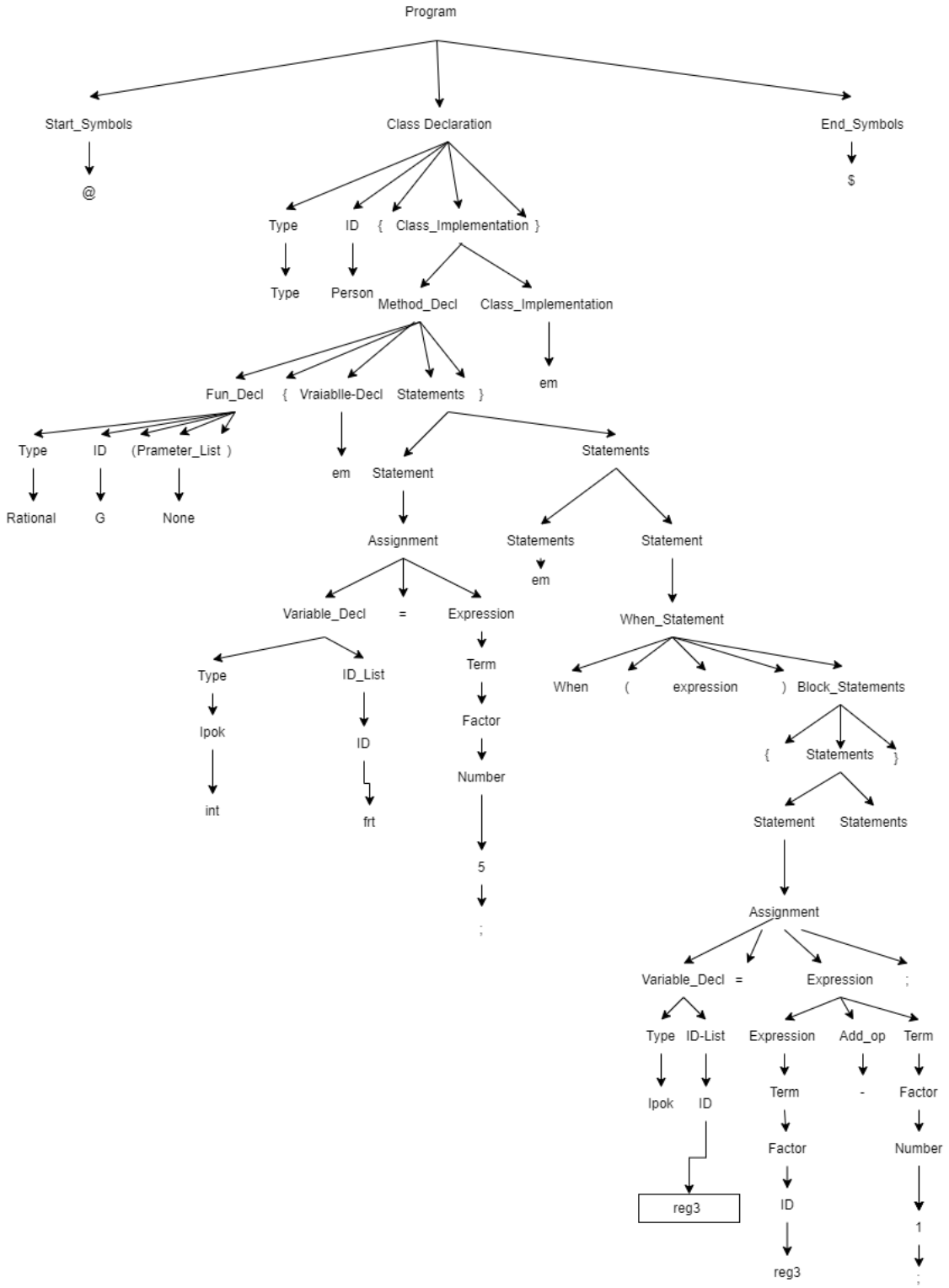

Logic Operators
em
148
&
149
&
150
|
151
|
152
~
153
Relational Operators
em
154
=
155
=
156
<
157
>
158
!
159
=
160
<
161
=
162
>
163
=
164
Assignment Operator
em
165
=
166
Access Operator
em
167
-
168
>
169
Braces
em
170
[
171
]
172
{
173
}
174
Quotation Mark
em
175
"
176
'
178
Inclusion
em
179
R
180
e
181
q
182
u
183
i
184
e
185
r
186
Comments
em
187
<
188
/
189
letter|digits
190
/
191
>
192
*
letter|digits
193
*
194
*
195
letter|digits
letter|digits

DFA

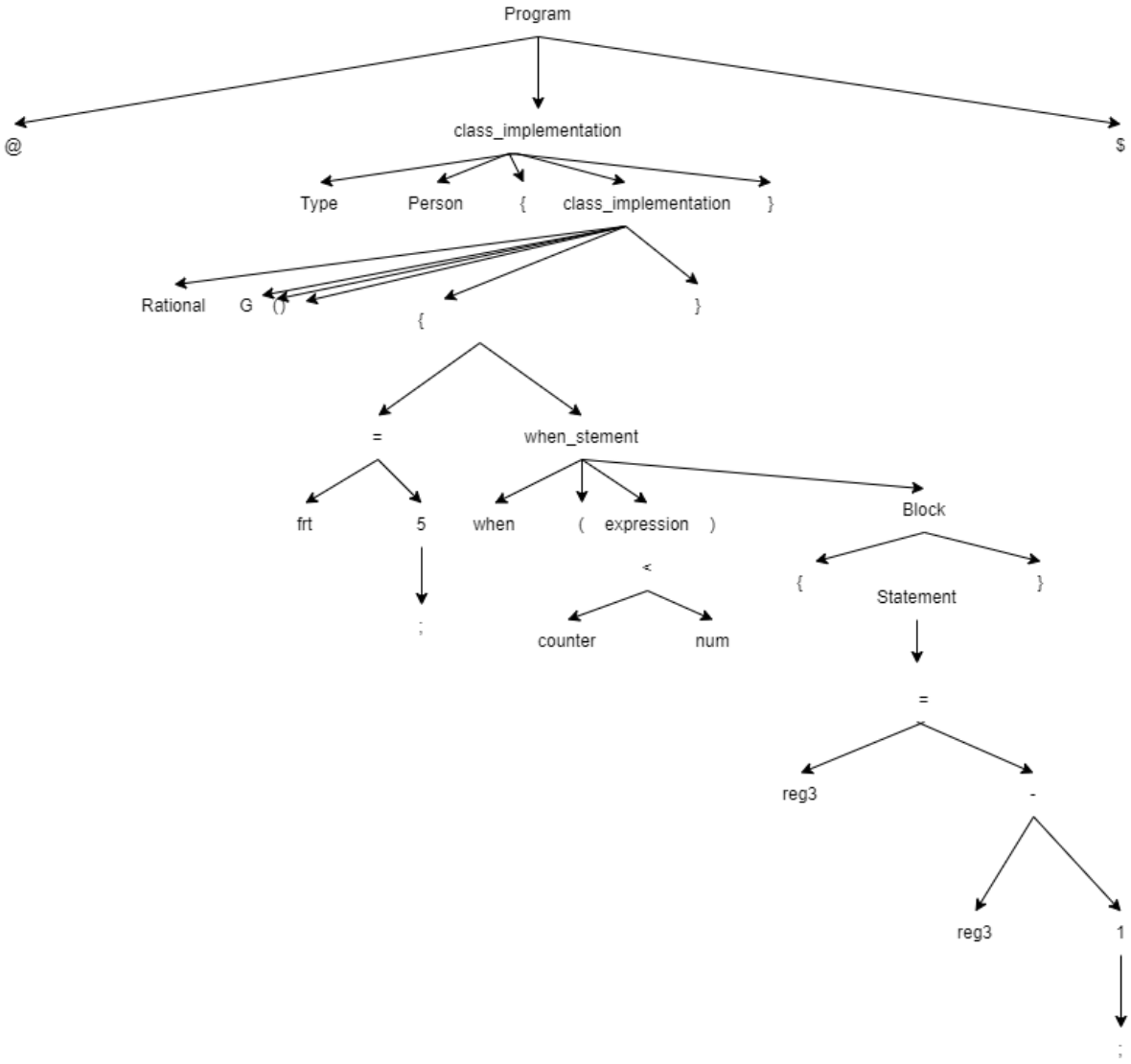

Start
0
@ | ^
1
T
2
y
3
p
4
e
5
I
6
p
7
o
8
k
9
f
10
n
11
f
12
e
13
r
14
f
15
E
16
l
17
s
18
e
19
n
20
d
21
l
22
h
23
i
24
s
25
S
26
l
27
p
28
o
29
k
30
f
31
e
32
q
33
u
34
e
35
n
36
c
37
e
38
c
39
r
40
a
41
n
42
a
43
n
44
C
45
r
46
a
47
f
48
o
49
n
50
d
51
l
52
t
53
l
54
o
55
n
56
o
57
f
58
V
59
a
60
l
61
u
62
l
63
e
64
s
65
s
66
R
67
a
68
t
69
l
70
o
71
n
72
a
73
l
74
e
75
q
76
u
77
l
78
r
79
e
80
s
81
p
82
o
83
n
84
d
85
w
86
i
87
t
88
h
89
H
90
o
91
w
92
e
93
v
94
e
95
r
96
W
97
h
98
e
99
n
100
Identifier
101
!
102
=
103
=
104
=
105
<
106
=
107
108
other
109
|
110
>
111
>
112
=
113
-
114
>
115
&
116
&
117
|
118
|
119
~
120
+
121
/
122
*
123
*
124
*
125
[
126
]
127
(
128
)
129
"
130
'
131
$
132
#
133


Start
0
@ | ^
1
T
5,138
y
6,138
p
7,138
e
8,138
I
25,49,138
p
26,50,138
o
27,51,136
k
28,54,138
f
53,138
n
11,138
f
12,138
e
13,138
r
14,138
f
16,138
E
20,81,138
l
21,138
s
22,138
e
23,138
n
82,138
d
83,138
t
84,138
h
85,138
i
86,138
s
87,138
S
29,39,54,115,120,138
i
30,55,138
p
31,56,128
o
32,57,138
k
33,58,138
f
60,138
e
41,138
q
42,138
u
43,138
e
44,138
n
45,138
c
46,138
e
47,138
c
120,115,138
r
120,116,138
a
120,117,138
n
120,118,138
a
122,138
n
122,138
C
35,126,138
r
36,138
a
37,138
f
38,138
o
82,138
n
83,138
d
84,138
i
85,138
t
86,138
i
87,138
o
87,138
n
87,138
o
87,138
f
87,138
V
62,138
a
63,138
l
64,138
u
65,138
l
66,138
e
67,138
s
68,138
s
69,138
R
72,176,103,138
a
73,138
t
74,138
i
75,138
o
76,138
n
77,138
a
78,138
l
79,138
e
177,102,138
q
178,138
u
179,138
i
180,138
r
181,138
e
182,138
s
104,138
p
105,138
o
106,138
n
107,138
d
108,138
w
109,138
i
110,138
t
111,138
h
112,138
H
89,138
o
90,138
w
91,138
e
92,138
v
93,138
e
94,138
r
95,138
W
97,138
h
98,138
e
99,138
n
100,138
Identifier
138
!
159
=
160
=
155,166
=
156
<
161,157,188
=
162
189
other
190
\
191
>
192
>
158,161
=
162
-
145,168
>
169
&
149
&
150
|
151
|
152
~
153
+
144
/
146
*
145,191
*
192
*
193
{
171
]
172
(
173
)
174
*
176
-
174
$
195
#
196

Parse Tree


Program
Start_Symbols
@
Class Declaration
Type
ID
{
Class_Implementation
}
Type
Person
Method_Decl
Class_Implementation
em
Fun_Decl
{
Vraiablle-Decl
Statements
}
Type
ID
(Prameter_List)
Rational
G
None
em
Statement
Statements
Assignment
Statements
Statement
em
Variable_Decl
=
Expression
When_Statement
Type
ID_List
Term
When
(
expression
)
Block_Statements
Ipok
ID
Factor
{
Statements
}
int
frt
Number
Statement
Statements
5
;
Assignment
Variable_Decl
=
Expression
;
Type
ID-List
Expression
Add_op
Term
Ipok
ID
Term
-
Factor
Factor
Number
reg3
ID
1
reg3
;

Abstract Parse Tree



Left Recursion Free Grammar Rules

(1) Program → Start-Symbols ClassDeclaration End-Symbols
(2) Start-Symbols → @| ^
(3) End-Symbols → $ | #
(4) ClassDeclaration→ Type ID {Class_Implementation} | Type ID Infer { Class_Implementation}
(5) Class_Implementation → Variable_Decl Class_Implementation| Method_Decl Class_Implementation | Comment Class_Implementation | require_command Class_Implementation| Func _Call Class_Implementation |em
(6) Method_Decl → Func Decl ;| Func Decl { Variable_Decl Statements }
(7) Func Decl →Type ID (ParameterList)
(8) Type → Ipok |Sipok |Craf |Sequence |Ipokf |Sipokf |Valueless |Rational
(9) ParameterList →em| None | Non-Empty List
(10) Non-Empty List→ Type ID | Non-Empty List , Type ID
Non-Empty List Type ID Non-EmptyList'
(11) Non-Empty List'Type ID Non-Empty List'|em
(12) Variable_Decl→ em | Type ID_List ; Variable_Decl | Type ID_List [ID] ; Variable_Decl
(13) ID_List →ID | ID_List , ID
ID_List ID ID_List'
(14) ID-List' ID ID_List'|em
(15) Statements→em | Statement Statements

(16) Statement→Assignment | If _Statement | However _Statement | when_Statement | Respondwith _ Statement | Endthis _Statement|Scanvalur (ID); | Print (Expression); |
(17) Assignment→ Variable_Decl = Expression;
(18) Func _Call → ID (Argument_List) ;
(19) Argument_List →em | NonEmpty_Argument_List
(20) NonEmpty_Argument_List →Expression | NonEmpty_Argument_List , Expression
NonEmpty_Argument_List Expression NonEmpty_Argument_List ‘
(21) NonEmty_Argument_List’ Expression NonEmpty_Argument_List’
(22) Block Statements→{ statements }
(23) If _Statement→ if (Condition _Expression) Block Statements | if (Condition _Expression) Block Statements else Block Statements
(24) Condition _Expression→ Condition |Condition Condition _Op Condition
(25) Condition _Op → && | ||
(26) Condition→ Expression Comparison _Op Expression
(27) Comparison _Op → == | != | > | >= | < | <=
(28) However _Statement → However (Condition _Expression) Block Statements
(29) when _Statement → when (expression ; expression ; expression) Block Statements
(30) Respondwith _Statement→ Respondwith Expression ; | return ID ;
(31) Endthis _Statement→ Endthis;

(32) Expression → Term |Expression Add_Op Term
ExpressionTerm Expression’
(33) Expession’ Add-Op Term Expression’
(34) Add_Op → + | -
(35) Term→Factor| Term Mul_Op Factor
TermFactor Term’
(36) Term’ Mul_Op Factor Term’
(37) Mul_Op→* | /
(38) Factor→ ID| Number
(39) Comment → | ***STR
(40) Require_command →Require(F_name.txt);
F_name →STR

First Grammar Rules

(1) First(Program) {@,^}
(2) First(Start-Symbols) {@,^}
(3) First(End-Symbols) {$,#}
(4) First(ClassDeclaration) {
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational }
(5) First(Class_Implementation) {
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID,em}
(6) First(Method_Decl){
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational}
(7) First(Func Decl){
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational }
(8) First(Type)
{Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational}
(9) First(ParameterList) {em, None,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational }
(10) First(Non-EmptyList) {
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational }
(11) First(Non-EmptyList’){em}
(12) First(Variable_Decl){em,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational }
(13) First(ID_List) {ID}
(14) First(ID_List’){em}
(15) First(Statements) { If,However, When,Respondwith,
Endthis,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,em}

(16) First(Statement) { If,However, When,Respondwith,
Endthis,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,em }
(17) First(Assignment) {
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,em }
(18) First(Func_Call) {ID}
(19) First(Argument_List){em,ID,Number}
(20) First(NonEmpty Argument List) {ID,Number}
(21) First(NonEmpty ArgumentList')){em,ID,Number}
(22) First(Block Statements){{}
(23) First(If_Statement){If}
(24) First(Condition_Expression){ID,Number}
(25) First(Condition_Op){&,|}
(26) First(Condition){ID,Number}
(27) First(Comparison_OP) {==,!-,>,>=,<,<=}
(28) First(However_Statement){However}
(29) First(When_Statement) {When}
(30) First(Respondwith_Statement) {Respondwith}
(31) First(Endthis_Statement){Endthis}
(32) First(Expression){ID,Number}
(33) First(Expression')){+,-}
(34) First(Add-Op){+,-}
(35) First(Term){ ID,Number }
(36) First(Term')){*,/}
(37) First(Mul_Op){*,/}
(38) First(Factor){ID,Number}
(39) First(Comment){</,***}
(40) First(Require_Command){Require}
(41) First(F_name){STR}

Follow Grammar Rules

(1) Follow(Program){$}
(2) Follow(Start-Symbols) {
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational}
(3) Follow(ClassDeclaration){#,$}
(4) Follow(End-Symbols){$}
(5) Follow(Type){ID}
(6) Follow(Class_Implementation){}}
(7) Follow(Variable_Decl) {
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID ,},=}
(8) Follow(Method_Decl){
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID,}}
(9) Follow(Comment){
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID,}}
(10) Follow(require_command){
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID,}}
(11) Follow(Func _Call){
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID,}}
(12) Follow(Func Decl){;,{}
(13) Follow(Variable_Decl){ { If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,

Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational,
<,***,require_Command, ID,}}
(14) Follow(Statements){}}
(15) Follow(ParameterList){)}
(16) Follow(Non-EmptyList){)}
(17) Follow(Non-EmptyList') {)}
(18) Follow(ID_List){;,[}
(19) Follow(ID_List') {;,[}
(20) Follow(Statement) { If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(21) Follow(Assignment){If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(22) Follow(If _Statement) {If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(23) Follow(However _Statement) { If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(24) Follow(when_Statement) {If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}

(25) Follow(Respondwith _ Statement)
{If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(26) Follow(Endthis _Statement) {If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(27) Follow(Scanvalur){(}
(28) Follow(print){(}
(29) Follow(Expression){),;,ID,Number,}
(30) Follow(Argument_List){)}
(31) Follow(NonEmpty_Argument_List){)}
(32) Follow(NonEmpty_Argument_List ‘){)}
(33) Follow(if){(}
(34) Follow(Condition_Expression){),}
(35) Follow(Block Statements){ If_Statement,However_Statement,
When_Statement,Respondwith_Statement, Endthis
_Statement,Scanvalur , Print,
Ipok,Sipok,Carf,Sequence,Ipokf,Sipokf,Valueless,Rational, }}
(36) Follow(Condition){), &&,||}
(37) Follow(Condition_Op){ ID,Number}
(38) Follow(Term) {),;,ID,Number, *,/}
(39) Follow(Add_Op) { ID,Number }
(40) Follow(Expression’) {),;,ID,Number,}
(41) Follow(Factor) { ID,Number, *,/ }
(42) Follow(Mul_Op) {ID,Number}

Parse Table

Excel FIle