Saint Petersburg Electrotechnical University "LETI"

The department MOEVM

The report on laboratory work №2

“Syntax analysis”

by discipline

“Development of linguistic processes”

Variant 19

Performed: A. Kondratev

Faculty CTI

Group 1381

Lecturer at the Department of MOEVM: Ivan Hodyrev

St. Petersburg

2015

In progress of the course work necessary to develop a programming language is a subset of the specified language, and the compiler from this language into an intermediate language, the type of which is determined by the option of individual tasks. Method of syntactic analysis (parsing) is also determined by the task.

The language must provide operations on variables and constants defined base types, as well as on the variables and the components of the derived type, which are defined for this assignment. The composition of operations should include both provided basic language, and listed in the job variant. In the language must be defined transaction type conversion when structural equivalence types or names. In the language has to be also possible to create user-defined types.

The language must allow the use of arithmetic expressions, in which may include constants and simple variables of basic types, the components of the structured type, parentheses, and signs of operations: addition, subtraction, multiplication, division. Priority of operations - an ordinary.

The language must allow the use of logical expressions of the relationship which may include parentheses and symbols of logic operations: AND, OR, NOT, and, in the case of a language in a Boolean constants and variables of this type. Priority of operations is usual.

Operations on variables of a structured type are defined for this job.

The composition of the operators of the language:

* assignment operator;
* operator input;
* output operator;
* composite operator;
* operator of unconditional transition;
* conditional statement, a condition in which a Boolean expression;
* cycle operator, a condition in which a Boolean expression.

The particular form of the operators is defined for this job.

The program in the input language can contain comments.

# Description of the source language.

## General information.

In the work describes the development of a compiler for the language is a subset of Pascal.

*Variant 19:*

* Basic language - Pascal.
* Basic types: integer, real, char.
* A structured type: character string.
* Operations on strings: the definition of the string length, string concatenation, replacing the substring, substring search, access to the substring.
* Additional requirements: functions, conditional assignment.
* The operator of the cycle - a precondition.
* The operator of overloading - allowed.
* Class of grammar - the grammar of operator precedence.
* The intermediate language - the triad.

## The formalism for describing the syntax.

The language syntax is described using modified Backus-Naur Form:

* Metalinguistic variables are composed of lowercase letters of the Russian alphabet, and underscores; the name of each variable is taken in <angle brackets>;
* Reserved words indicated by the words of the English language, consisting of lowercase Latin letters;
* In [brackets] are concluded elements that may be absent;
* Repeated zero or more elements are in {curly braces};

# The full syntax of the language in Backus-Naur Form.

|  |
| --- |
| **<program>**::= ['program' < identifier>;] {<block\_definitions\_constant>|<block\_definitions\_label>|   < block\_definitions \_variables >|< block\_definitions \_function>} < composite\_operator > '.'  < **block\_definitions \_function**>::= < operator\_function >|< operator\_ overloading>  < **block\_definitions \_variables** >::=’var’ < variable\_description > {< variable\_description >}  **<** **variable\_description >**::=< identifier >{','< identifier >}': '< simple\_type >';'  < **block\_definitions\_constant** >::=’const’ <constant\_description> ‘;’ { < constant\_description > ‘;’ }  **<** **constant\_description>**::=< identifier >'='<constant>  < **block\_definitions\_label** >::= ‘label’ < identifier > {‘,’ <identifier>} ‘;’  **<** **simple\_type>**::='integer'|'real'|’boolean’|’char’|’string’  **<** **identifier >**::=<letter><sequence\_of\_letters\_and\_ digits> **<letter>**::='\_'|'a'|'b'|'c'|'d'|'e'|'f'|'g'|'h'|'i'|'j'|'k'|'l'|'m'|'n'|'o'|'p'|'q'|'r'|'s'|'t'|'u'|'v'|'w'|'x'|'y'|'z'|  |'A'|'B'|'C'|'D'|'E'|'F'|'G'|'H'|'I'|'J'|'K'|'L'|'M'|'N'|'O'|'P'|'Q'|'R'|'S'|'T'|'U'|'V'|'W'|'X'|'Y'|'Z'  **<digit>**::='0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'  < **sequence\_of\_letters\_and\_ digits** >::={<letter>|<digit>}  **<arithmetic\_expression**>::=[<sign>]< therm >|< arithmetic\_expression ><sign><therm>  **< therm>::=<** multiplier>|< therm >’\*’< multiplier >|< therm >’/’< multiplier >  **<multiplier**>::='('<arithmetic\_expression>')'|<identifier>|<decimal\_constant>|< real\_constant>  **<sign>::='+'|'-'**  **<** **logical\_expressions>**::=<logical\_therm>{’or ’< logical\_therm>}  **<** **logical\_therm >**::=< logical\_therm>’ and ’<logical\_multiplier>|<logical\_multiplier>  **<** **logical\_multiplier>**::=< identifier>|< arithmetic\_expression><ratio><arithmetic\_expression>|   <string\_constant><ratio><string\_constant>|’not’<logical\_multiplier>|’(‘<logical\_expressions >’)’  **<** **ratio >**::='='|'<'|'>'|'<='|'>=’  **<constant>**::=<decimal\_constant>|<real\_constant>|<char\_constant>|< Boolean\_constant>|< string\_constant >  **<** **decimal\_constant >**::=[<sign>]<sequence\_of\_digits>  < **sequence\_of\_digits** >::=<digit>{<digit>}  **<** **real\_constant >**::=< fractional\_constant >[< exponential\_part >]  < **fractional\_constant** >::=< sequence\_of\_digits >[.< sequence\_of\_digits >]  < **exponential\_part** >::=’e’[<sign>]< sequence\_of\_digits>|’E’[<sign>] < sequence\_of\_digits >  **<** **char\_constant >**::='''[<letter>|<digit>]'''  < **Boolean\_constant** >::=’true’|’false’  **<** **string\_constant >**::='''< sequence\_of\_letters\_and\_ digits >'''  **<operator>**::= { [<identifier>':']< unlabeled\_operator > }  **<** **unlabeled\_operator >**::= <assignment\_operator>|< operator\_of\_unconditional\_transition >| <output\_operator>|   <input\_operator>|<composite\_operator>|<conditional\_operator>|<while\_loop>|<operator\_function>| < operator\_conditional\_assignment >|<operator\_ overloading>|ε  **<assignment\_operator**>::=<identifier>':='<arithmetic\_expression>|<logical\_expression>|<constant>|  < operator\_string\_concatenation>|< operator\_replacement\_substring >|< operator\_access\_substring >  **<** **operator\_of\_unconditional\_transition >**::='goto' <identifier>  **<output\_operator**>::='write('<message> {','<message>} ')'  **<message**>::= <arithmetic\_expression>|<logical\_expression>|<string\_constant**>**  **<input\_operator>**::='read(' <identifier> {','<identifier>} ')'  **<composite\_operator>**::= 'begin' {<operator> ';'} 'end'  **<** **conditional\_operator >**::='if '<logical\_expression>' then '<operator>|<composite\_operator> [' else  '<operator>|<composite\_operator>]  **<while\_loop>**::='while '<logical\_expression> ' do ' <operator>|<composite\_operator>  **<operator\_length\_string>**::='length('<identifier>|<string\_constant>')'  **<operator\_string\_concatenation>**::='concat('<identifier>|<string\_constant>', '<identifier>|< string\_constant >')'  **<operator\_replacement\_substring>**::='replace('<identifier>|<string\_constant>', '<identifier>|<string\_constant>',   '<identifier >|< string\_constant >')'  **<search\_operator\_substring>**::='search('<identifier>|<string\_constant>', '<identifier>|<string\_constant>')'  **<operator\_access\_substring >**::='substr('<identifier>|<string\_constant>', '<identifier>|<sequence\_of\_digits>',   '< identifier>|<sequence\_of\_digits >')'  <**operator\_function**>::=<function\_header> [<block\_definitions \_variables >] <composite\_operate> ‘;’  < **function\_header**>::=’function’ <identifier> ‘:’ <simply\_type> ‘;’  < **operator\_conditional\_assignment** >::= ‘if’ <logical\_expression> 'then' <assignment\_operator> ['else'   <assignment\_operator >]  < **operator\_ overloading** >::= ‘operator’ < overloaded \_operations > ‘(‘<description\_overloaded\_variables > ’)’   <identifier> ‘:’ <simple\_type> ‘;’ <composite\_operator > ‘;’  < **description\_overloaded\_variables \_1**>::= < identifier> ‘:’ < simple\_type > ‘;’ < identifier> ‘:’ < simple\_type >  < **description\_overloaded\_variables \_2**>::= < identifier> ‘,’ < identifier> ‘:’ < simple\_type >  < **description\_overloaded\_variables** >::=< description\_overloaded\_variables \_1> |   < description\_overloaded\_variables \_2>  < **overloaded \_operations**> ::= <ratio>|’+’|’-’|’**\***’|’/’ |

# Description of the syntax analysis.

## Construction of CFG(context-free grammar) input language.

For the construction of CFG input language is necessary:

* replace metalinguistic variables BNF notation of non-terminal symbols, using short names;
* terminal symbols as used tokens;
* metalinguistic symbol ":: =" to replace the symbol «→»;
* replace one metalinguistic formula n alternatives by n grammar rules with the same symbol on the left of inference rules;
* exclude metalinguistic symbols {} and [] to include the rules of grammar to recursive rules and ε-rule.

The correspondences between the non-terminal symbols of the grammar and meta linguistic variables Backus-Naur Form:

|  |  |  |  |
| --- | --- | --- | --- |
| <program> | **PROG** | < sequence\_of\_letters\_and\_ digits > | **NLN** |
| < block\_definitions \_variables > | **VARS** | <operator> | **OP** |
| < variable\_description > | **VAR** | < unlabeled\_operator > | **UMOP** |
| < block\_definitions\_constant> | **CONSTS** | < assignment\_operator > | **AOP** |
| < constant\_description > | **CONST** | < operator\_of\_unconditional\_transition> | **GOP** |
| < block\_definitions\_labels> | **LABELS** | <output\_operator> | **WOP** |
| <simple\_type> | **TYPE** | <massage> | **MSG** |
| <identifier> | **ID** | <input\_operator> | **ROP** |
| <letter> | **L** | <composite\_operator> | **COP** |
| <digit> | **N** | < conditional\_operator > | **IFOP** |
| <arithmetic\_expression> | **AEXP** | < while\_loop > | **FOROP** |
| <therm> | **TERM** | < operator\_length\_string > | **LENOP** |
| < multiplier > | **MULT** | < operator\_string\_concatenation > | **CONOP** |
| <sign> | **SIGN** | < operator\_replacement\_substring > | **REPOP** |
| < logical\_expression > | **LEXP** | < search\_operator\_substring > | **SEOP** |
| <logical\_therm> | **LTERM** | < operator\_access\_substring > | **SUBOP** |
| <logical\_multiplier> | **LMULT** | < operator\_function > | **FUNOP** |
| <ratio> | **REL** | < function\_header > | **HFUN** |
| <constant> | **C** | < operator\_conditional\_assignment > | **IFAOP** |
| < decimal\_constant > | **DEC** | < operator\_ overloading > | **OPOP** |
| < sequence\_of\_digits > | **NN** | < description\_overloaded\_variables \_1> | **OVAR1** |
| < real\_constant > | **RC** | < description\_overloaded\_variables \_2> | **OVAR2** |
| < fractional\_constant > | **DC** | < description\_overloaded\_variables > | **OVAR** |
| < exponential\_part > | **EXP** | <overloaded \_operations> | **OOP** |
| < char\_constant > | **CC** | <block\_definitions\_function> | **FUNCS** |
| < Boolean\_constant > | **BC** |  |  |
| < string\_constant > | **SC** |  |  |

All grammars of syntax analysis:

Grammar programs G0 = <T,N,PROG,R>

|  |
| --- |
| N = {PROG, ID, COP, *CLV*, *OPROG, OCOP*} |
| T = {\_PROG, ., ;} |
| R = {   |  |  |  | | --- | --- | --- | | PROG →\_PROG ID ; OPROG  PROG → OPROG | OCOP→COP.  OPROG → OCOP | OPROG → CLV; OCOP |   } |

The grammar of block definitions of variables G1 = <T,N,VARS,R>

|  |
| --- |
| N = {VARS, VAR, ID, TYPE, *RVAR*, *RID*} |
| T = {\_VAR, ,, ;, :} |
| R = {   |  |  |  | | --- | --- | --- | | VARS→\_VAR RVAR  RVAR→VAR ; RVAR | RVAR→VAR  VAR→RID : TYPE | RID→ ID , RID  RID→ ID |   } |

The grammar of block definitions of constants G2 = <T,N,CONSTS,R>

|  |
| --- |
| N = {CONSTS, CONST, ID, C, *RCONST*} |
| T = {\_CONST, =, ;} |
| R = {   |  |  |  | | --- | --- | --- | | CONSTS →\_CONST CONST ; RCONST  CONSTS →\_CONST CONST  RCONST → CONST ; RCONST | RCONST → CONST  CONST→ID = C |  |   } |

The grammar of block definitions of labels G3 = <T,N,LABELS,R>

|  |
| --- |
| N = {LABELS, *RID*} |
| T = {\_LABEL} |
| R = {   |  |  |  | | --- | --- | --- | | LABELS →\_LABEL RID |  |  |   } |

The grammar of block definitions of functions G4 = <T,N,FUNCS,R>

|  |
| --- |
| N = {FUNCS, FUNOP, OPOP, HFUN, VARS, COP, ID, TYPE } |
| T = {\_FUNC, :, ;} |
| R = {   |  |  |  | | --- | --- | --- | | FUNCS→FUNOP  FUNOP → HFUN ; COP ;  FUNOP → HFUN ; VARS ; COP ; | FUNCS→OPOP  HFUN → \_FUNC ID : TYPE |  |   } |

The grammar of simple type G5 = <T,N,TYPE,R>

|  |
| --- |
| N = {TYPE} |
| T = {\_INT, \_REAL, \_BOOL, \_CHAR, \_STRING} |
| R = {   |  |  |  | | --- | --- | --- | | TYPE→\_INT  TYPE→\_REAL | TYPE→\_BOOL  TYPE→\_CHAR | TYPE→\_STRING |   } |

The grammar of identifier G6 = <T,N,ID,R>

|  |
| --- |
| N = {ID} |
| T = { \_ID, ( , )} |
| R = {   |  |  |  | | --- | --- | --- | | ID→\_ID | ID→\_ID ( ) |  |   } |

The grammar of arithmetic expression G7 = <T,N,AEXP,R>

|  |
| --- |
| N = {AEXP, TERM, MULT, ID, LENOP, SEOP } |
| T = {\*, /, (, ), +, -, \_NUM, \_RC} |
| R = {   |  |  |  | | --- | --- | --- | | AEXP → TERM  AEXP → + TERM  AEXP → - TERM  AEXP → AEXP + TERM  AEXP → AEXP - TERM  TERM → MULT | TERM → TERM \* MULT  TERM → TERM / MULT  MULT → ( AEXP ) | MULT → ID  MULT → \_NUM  MULT → \_RC  MULT → LENOP  MULT → SEOP |   } |

The grammar of logical expression G8 = <T,N,LEXP,R>

|  |
| --- |
| N = {LEXP, LTERM, LMULT, ID, AEXP, *RLTERM*} |
| T = {\_OR, \_AND, \_NOT, \_REL,\_STR, (, )} |
| R = {   |  |  |  | | --- | --- | --- | | LEXP → LTERM  LEXP → LTERM \_OR RLTERM  RLTERM → LTERM \_OR RLTERM  RLTERM → LTERM  LTERM → LTERM \_AND LMULT | LTERM → LMULT  LMULT → ID  LMULT → AEXP \_REL AEXP  LMULT → \_STR \_REL\_STR | LMULT → \_NOT LMULT  LMULT → ( LEXP ) |   } |

The grammar of constant G9 = <T,N,C,R>

|  |
| --- |
| N = {C, DEC, RC, CC, BC, SC, N, DC, L} |
| T = {., \_TRUE, \_FALSE, \_NUM, \_STR, +, -, \_E, \_CC, \_RC } |
| R = {   |  |  |  | | --- | --- | --- | | C→ DEC  C → RC  C → CC  C → BC  C → SC  DEC → + \_NUM  DEC → - \_NUM | DEC → \_NUM  RC → DC  RC → \_RC  DC → \_NUM  DC → \_NUM . \_NUM | CC → \_CC  BC → \_TRUE  BC → \_FALSE  SC → \_STR |   } |

The grammar of operator G10 = <T,N,C,R>

|  |
| --- |
| N = {OP, ID, UMOP, AOP, GOP, WOP, ROP, COP, IFOP, FOROP, FUNOP, CONOP, NEWOP, IFAOP, OPOP, REPOP, SEOP, SUBOP, *RECUROP* } |
| T = {:, ;} |
| R = {   |  |  |  | | --- | --- | --- | | OP → RECUROP  RECUROP → ID : UMOP ; RECUROP  RECUROP → UMOP ; RECUROP  RECUROP → UMOP | UMOP → AOP|GOP|WOP|ROP|COP|IFOP|FOROP|CONOP|REPOP|SEOP|SUBOP|FUNOP|NEWOP|IFAOP|OPOP |  |   } |

The grammar of assignment operator G11 = <T,N,AOP,R>

|  |
| --- |
| N = {AOP, ID, AEXP, LEXP, C, CONOP, SUBOP, REPOP, *ALEXP*} |
| T = {\_EQ} |
| R = {   |  |  |  | | --- | --- | --- | | AOP → ID \_EQ ALEXP  ALEXP → C  ALEXP → SUBOP | ALEXP → AEXP  ALEXP → CONOP | ALEXP → LEXP  ALEXP → REPOP |   } |

The grammar of operator of unconditional transition G12 = <T,N,GOP,R>

|  |
| --- |
| N = {GOP, ID } |
| T = {\_GOTO} |
| R = {   |  |  |  | | --- | --- | --- | | GOP → \_GOTO ID |  |  |   } |

The grammar of output operator G13 = <T,N,WOP,R>

|  |
| --- |
| N = {WOP, MSG, AEXP, LEXP, *RMSG*} |
| T = {\_WRITE,\_STR, (, ), ,} |
| R = {   |  |  |  | | --- | --- | --- | | WOP → \_WRITE ( RMSG )  RMSG → MSG , RMSG | RMSG → MSG  MSG → AEXP | MSG → LEXP  MSG → \_STR |   } |

The grammar of input operator G14 = <T,N,ROP,R>

|  |
| --- |
| N = {ROP, ID, *RID*} |
| T = {\_READ, (, ), ,} |
| R = {   |  |  |  | | --- | --- | --- | | ROP → \_READ ( RID )  RID->ID,RID  RID->ID |  |  |   } |

The grammar of composite operator G15 = <T,N,COP,R>

|  |
| --- |
| N = {COP, OP, *ZOMOP*} |
| T = {\_BEGIN, \_END, ;} |
| R = {   |  |  |  | | --- | --- | --- | | COP → \_BEGIN ZOMOP \_END  COP → \_BEGIN \_END | ZOMOP → OP ; ZOMOP | ZOMOP → OP ; |   } |

The grammar of conditional\_operator G16 = <T,N,IFOP,R>

|  |
| --- |
| N = {IFOP, LEXP, OP, COP, *OCOP*} |
| T = {\_IF, \_THEN, \_ELSE} |
| R = {   |  |  |  | | --- | --- | --- | | IFOP → \_IF LEXP \_THEN OCOP  IFOP → \_IF LEXP \_THEN OCOP \_ELSE OCOP | OCOP → COP  OCOP → OP |  |   } |

The grammar of while loop operator G17 = <T,N,FOROP,R>

|  |
| --- |
| N = {FOROP, LEXP, OP, COP, *OCOP*} |
| T = {\_WHILE, \_DO} |
| R = {   |  |  |  | | --- | --- | --- | | FOROP → \_WHILE LEXP \_DO OCOP |  |  |   } |

The grammar of operator length string G18 = <T,N,LENOP,R>

|  |
| --- |
| N = {LENOP, ID, SC, *IDSC*} |
| T = {\_LEN, (, )} |
| R = {   |  |  |  | | --- | --- | --- | | LENOP → \_LEN ( IDSC ) | IDSC → ID | IDSC → SC |   } |

The grammar of operator of string concatenation G19 = <T,N,CONOP,R>

|  |
| --- |
| N = {CONOP, ID, SC, *IDSC*} |
| T = {\_CONC, (, ), ,} |
| R = {   |  |  |  | | --- | --- | --- | | CONOP → \_CONC ( IDSC , IDSC ) |  |  |   } |

The grammar of operator of replacement substring to string G20 = <T,N,REPOP,R>

|  |
| --- |
| N = {REPOP, ID, SC, *IDSC*} |
| T = {\_REPL, (, ), ,} |
| R = {   |  |  |  | | --- | --- | --- | | REPOP → \_REPL ( IDSC , IDSC, IDSC ) |  |  |   } |

The grammar of operator of search substring in the string G21 = <T,N,SEOP,R>

|  |
| --- |
| N = {SEOP, ID, SC, *IDSC*} |
| T = {\_SRCH, (, ), ,} |
| R = {   |  |  |  | | --- | --- | --- | | SEOP → \_SRCH ( IDSC , IDSC ) |  |  |   } |

The grammar of operator of access substring to string G22 = <T,N,SUBOP,R>

|  |
| --- |
| N = {SUBOP, ID, SC, *IDSC, IDNN*} |
| T = {\_SUBS,\_NUM, (, ), ,} |
| R = {   |  |  |  | | --- | --- | --- | | SUBOP → \_SUBS ( IDSC , IDNN, IDNN ) | IDNN → ID | IDNN → \_NUM |   } |

The grammar of program G23 = <T,N,CLV,R>

|  |
| --- |
| N = {CONSTS, LABELS, VARS, FUNCS, *CLV*} |
| T = { ; } |
| R = {   |  |  |  | | --- | --- | --- | | CLV → CONSTS  CLV → CONSTS ; CLV  CLV → LABELS | CLV→VARS  CLV→VARS ; CLV  CLV → LABELS ; CLV | CLV→FUNCS  CLV→FUNCS ; CLV |   } |

The grammar of operator conditional assignment G25 = <T,N,IFAOP,R>

|  |
| --- |
| N = {IFAOP, LEXP, AOP, *OEAOP*} |
| T = {\_IF, \_THEN, \_ELSE} |
| R = {   |  |  |  | | --- | --- | --- | | IFAOP → \_IF LEXP \_THEN AOP \_ELSE AOP  IFAOP → \_IF LEXP \_THEN AOP |  |  |   } |

The grammar of operator of overloading G26 = <T,N,OPOP,R>

|  |
| --- |
| N = {OPOP, OOP, OVAR, ID, TYPE, COP, OVAR1, OVAR2, REL} |
| T = {\_OPER, (, ), :, ;, ,, +, -, \*, /, \_REL} |
| R = {   |  |  |  | | --- | --- | --- | | OPOP → \_OPER OOP ( OVAR ) ID : TYPE ; COP ;  OVAR1 → ID : TYPE ; ID : TYPE  OVAR2 → ID , ID : TYPE | OVAR → OVAR1  OVAR → OVAR2  OOP → REL|+|-|\*|/ | REL → \_REL |   } |

Italics are non-terminals which are subsidiary in removing symbols {} and [] from the BNF.

## Definition of the class CFG of the input language.

Each of the constructed CFG in the previous point belongs to the class of grammars of operator precedence.

## DMP converter.

The control table:

File : "G0.TXT" .

NON-TERMINALS: [CLV, COP, ID, OCOP, OPROG, PROG]

TERMINALS: [., ;, \_prog]

START SYMBOL: PROG

RULES:

(0) PROG -> \_prog ID ; OPROG

(1) PROG -> OPROG

(2) OPROG -> OCOP

(3) OPROG -> CLV ; OCOP

(4) OCOP -> COP .

The grammar of operator precedence

Table of transfer and convolution

| €

-------------------------+---------

# [\_prog, PROG, ;, PROG] | 0, PROG

-------------------------+---------

; [PROG, ;, PROG] | 3, PROG

-------------------------+---------

# [PROG, ;, PROG] | 3, PROG

-------------------------+---------

; [PROG, .] | 4, PROG

-------------------------+---------

# [PROG, .] | 4, PROG

| . | ; | \_prog | $

-----------+---+---+-------+---

[., C] | | | | R

-----------+---+---+-------+---

[;, C] | S | S | | R

-----------+---+---+-------+---

[\_prog, C] | | S | |

-----------+---+---+-------+---

[#] | S | S | S |

-----------+---+---+-------+---

[#, PROG] | S | S | S | A

The relationship of precedence

| . | ; | \_prog | $

------+------+------+-------+------

. | | | | [.>]

------+------+------+-------+------

; | [<.] | [<.] | | [.>]

------+------+------+-------+------

\_prog | | [=.] | |

------+------+------+-------+------

# | [<.] | [<.] | [<.] |

End.

File : "G1.TXT" .

NON-TERMINALS: [ID, RID, RVAR, TYPE, VAR, VARS]

TERMINALS: [,, :, ;, \_var]

START SYMBOL: VARS

RULES:

(0) VARS -> \_var RVAR

(1) RVAR -> VAR ; RVAR

(2) RVAR -> VAR

(3) VAR -> RID : TYPE

(4) RID -> ID , RID

(5) RID -> ID

The grammar of operator precedence

Table of transfer and convolution

| €

---------------------+---------

# [\_var, VARS] | 0, VARS

---------------------+---------

; [VARS, ;, VARS] | 1, VARS

---------------------+---------

\_var [VARS, ;, VARS] | 1, VARS

---------------------+---------

; [VARS, :, VARS] | 3, VARS

---------------------+---------

\_var [VARS, :, VARS] | 3, VARS

---------------------+---------

, [VARS, ,, VARS] | 4, VARS

---------------------+---------

; [VARS, ,, VARS] | 4, VARS

---------------------+---------

\_var [VARS, ,, VARS] | 4, VARS

| , | : | ; | \_var | $

----------+---+---+---+------+---

[,, C] | S | R | | |

----------+---+---+---+------+---

[:, C] | | | R | | R

----------+---+---+---+------+---

[;, C] | S | S | S | | R

----------+---+---+---+------+---

[\_var, C] | S | S | S | | R

----------+---+---+---+------+---

[#] | | | | S |

----------+---+---+---+------+---

[#, VARS] | | | | S | A

The relationship of precedence

| , | : | ; | \_var | $

-----+------+------+------+------+------

, | [<.] | [.>] | | |

-----+------+------+------+------+------

: | | | [.>] | | [.>]

-----+------+------+------+------+------

; | [<.] | [<.] | [<.] | | [.>]

-----+------+------+------+------+------

\_var | [<.] | [<.] | [<.] | | [.>]

-----+------+------+------+------+------

# | | | | [<.] |

End.

File : "G2.TXT" .

NON-TERMINALS: [C, CONST, CONSTS, ID, RCONST]

TERMINALS: [;, =, \_const]

START SYMBOL: CONSTS

RULES:

(0) CONSTS -> \_const CONST ; RCONST

(1) CONSTS -> \_const CONST

(2) RCONST -> CONST ; RCONST

(3) RCONST -> CONST

(4) CONST -> ID = C

The grammar of operator precedence

Table of transfer and convolution

| €

------------------------------+-----------

# [\_const, CONSTS, ;, CONSTS] | 0, CONSTS

------------------------------+-----------

# [\_const, CONSTS] | 1, CONSTS

------------------------------+-----------

; [CONSTS, ;, CONSTS] | 2, CONSTS

------------------------------+-----------

; [CONSTS, =, CONSTS] | 4, CONSTS

------------------------------+-----------

\_const [CONSTS, =, CONSTS] | 4, CONSTS

| ; | = | \_const | $

------------+---+---+--------+---

[;, C] | S | S | | R

------------+---+---+--------+---

[=, C] | R | | | R

------------+---+---+--------+---

[\_const, C] | S | S | | R

------------+---+---+--------+---

[#] | | | S |

------------+---+---+--------+---

[#, CONSTS] | | | S | A

The relationship of precedence

| ; | = | \_const | $

-------+------+------+--------+------

; | [<.] | [<.] | | [.>]

-------+------+------+--------+------

= | [.>] | | | [.>]

-------+------+------+--------+------

\_const | [=.] | [<.] | | [.>]

-------+------+------+--------+------

# | | | [<.] |

End.

File : "G3.TXT" .

NON-TERMINALS: [ID, LABELS, RID]

TERMINALS: [,, \_label]

START SYMBOL: LABELS

RULES:

1. LABELS -> \_label RID
2. RID -> ID , RID
3. RID -> ID

The grammar of operator precedence

Table of transfer and convolution

| €

---------------------------+-----------

# [\_label, LABELS] | 0, LABELS

---------------------------+-----------

, [LABELS, ,, LABELS] | 1, LABELS

---------------------------+-----------

\_label [LABELS, ,, LABELS] | 1, LABELS

| , | \_label | $

------------+---+--------+---

[,, C] | S | | R

------------+---+--------+---

[\_label, C] | S | | R

------------+---+--------+---

[#] | | S |

------------+---+--------+---

[#, LABELS] | | S | A

The relationship of precedence

| , | \_label | $

-------+------+--------+------

, | [<.] | | [.>]

-------+------+--------+------

\_label | [<.] | | [.>]

-------+------+--------+------

# | | [<.] |

End.

File : "G4.TXT" .

NON-TERMINALS: [COP, FUNCS, FUNOP, HFUN, ID, OPOP, TYPE, VARS]

TERMINALS: [:, ;, \_func]

START SYMBOL: FUNCS

RULES:

(0) FUNCS -> FUNOP

(1) FUNCS -> OPOP

(2) FUNOP -> HFUN ; COP ;

(3) FUNOP -> HFUN ; VARS ; COP ;

(4) HFUN -> \_func ID : TYPE

The grammar of operator precedence

Table of transfer and convolution

| €

---------------------------------+----------

# [FUNCS, ;, FUNCS, ;] | 2, FUNCS

---------------------------------+----------

# [FUNCS, ;, FUNCS, ;, FUNCS, ;] | 3, FUNCS

---------------------------------+----------

# [\_func, FUNCS, :, FUNCS] | 4, FUNCS

| : | ; | \_func | $

-----------+---+---+-------+---

[:, C] | | R | |

-----------+---+---+-------+---

[;, C] | | S | | R

-----------+---+---+-------+---

[\_func, C] | S | | |

-----------+---+---+-------+---

[#] | | S | S |

-----------+---+---+-------+---

[#, FUNCS] | | S | S | A

The relationship of precedence

| : | ; | \_func | $

------+------+------+-------+------

: | | [.>] | |

------+------+------+-------+------

; | | [=.] | | [.>]

------+------+------+-------+------

\_func | [=.] | | |

------+------+------+-------+------

# | | [<.] | [<.] |

End.

File : "G5.TXT" .

NON-TERMINALS: [TYPE]

TERMINALS: [\_bool, \_char, \_int, \_real, \_string]

START SYMBOL: TYPE

RULES:

(0) TYPE -> \_int

(1) TYPE -> \_real

(2) TYPE -> \_bool

(3) TYPE -> \_char

(4) TYPE -> \_string

The grammar of operator precedence

Table of transfer and convolution

| €

------------+---------

# [\_int] | 0, TYPE

------------+---------

# [\_real] | 1, TYPE

------------+---------

# [\_bool] | 2, TYPE

------------+---------

# [\_char] | 3, TYPE

------------+---------

# [\_string] | 4, TYPE

| \_bool | \_char | \_int | \_real | \_string | $

-------------+-------+-------+------+-------+---------+---

[\_bool, C] | | | | | | R

-------------+-------+-------+------+-------+---------+---

[\_char, C] | | | | | | R

-------------+-------+-------+------+-------+---------+---

[\_int, C] | | | | | | R

-------------+-------+-------+------+-------+---------+---

[\_real, C] | | | | | | R

-------------+-------+-------+------+-------+---------+---

[\_string, C] | | | | | | R

-------------+-------+-------+------+-------+---------+---

[#] | S | S | S | S | S |

-------------+-------+-------+------+-------+---------+---

[#, TYPE] | S | S | S | S | S | A

The relationship of precedence

| \_bool | \_char | \_int | \_real | \_string | $

--------+-------+-------+------+-------+---------+------

\_bool | | | | | | [.>]

--------+-------+-------+------+-------+---------+------

\_char | | | | | | [.>]

--------+-------+-------+------+-------+---------+------

\_int | | | | | | [.>]

--------+-------+-------+------+-------+---------+------

\_real | | | | | | [.>]

--------+-------+-------+------+-------+---------+------

\_string | | | | | | [.>]

--------+-------+-------+------+-------+---------+------

# | [<.] | [<.] | [<.] | [<.] | [<.] |

File : "G6.TXT" .

NON-TERMINALS: [ID]

TERMINALS: [\_id]

START SYMBOL: ID

RULES:

(0) ID -> \_id

The grammar of operator precedence

Table of transfer and convolution

| €

--------+-------

# [\_id] | 0, ID

| \_id | $

---------+-----+---

[\_id, C] | | R

---------+-----+---

[#] | S |

---------+-----+---

[#, ID] | S | A

The relationship of precedence

| \_id | $

----+------+------

\_id | | [.>]

----+------+------

# | [<.] |

End.

File : "G7.TXT" .

NON-TERMINALS: [AEXP, ID, LENOP, MULT, SEOP, TERM]  
TERMINALS: [(, ), \*, +, -, /, \_num, \_rc]  
START SYMBOL: AEXP  
 RULES:  
(0) AEXP -> TERM

(1) AEXP -> + TERM

(2) AEXP -> - TERM

(3) AEXP -> AEXP + TERM

(4) AEXP -> AEXP - TERM

(5) TERM -> MULT

(6) TERM -> TERM \* MULT

(7) TERM -> TERM / MULT

(8) MULT -> ( AEXP )

(9) MULT -> ID

(10) MULT -> \_num

(11) MULT -> \_rc

(12) MULT -> LENOP

(13) MULT -> SEOP  
The grammar of operator precedence

Table of transfer and convolution  
 | €

------------------+----------

( [+, AEXP] | 1, AEXP

------------------+----------

# [+, AEXP] | 1, AEXP

------------------+----------

( [-, AEXP] | 2, AEXP

------------------+----------

# [-, AEXP] | 2, AEXP

------------------+----------

( [AEXP, +, AEXP] | 3, AEXP

------------------+----------

# [AEXP, +, AEXP] | 3, AEXP

------------------+----------

( [AEXP, -, AEXP] | 4, AEXP

------------------+----------

# [AEXP, -, AEXP] | 4, AEXP

------------------+----------

( [AEXP, \*, AEXP] | 6, AEXP

------------------+----------

+ [AEXP, \*, AEXP] | 6, AEXP

------------------+----------

- [AEXP, \*, AEXP] | 6, AEXP

------------------+----------

# [AEXP, \*, AEXP] | 6, AEXP

------------------+----------

( [AEXP, /, AEXP] | 7, AEXP

------------------+----------

+ [AEXP, /, AEXP] | 7, AEXP

------------------+----------

- [AEXP, /, AEXP] | 7, AEXP

------------------+----------

# [AEXP, /, AEXP] | 7, AEXP

------------------+----------

( [(, AEXP, )] | 8, AEXP

------------------+----------

\* [(, AEXP, )] | 8, AEXP

------------------+----------

+ [(, AEXP, )] | 8, AEXP

------------------+----------

- [(, AEXP, )] | 8, AEXP

------------------+----------

/ [(, AEXP, )] | 8, AEXP

------------------+----------

# [(, AEXP, )] | 8, AEXP

------------------+----------

( [\_num] | 10, AEXP

------------------+----------

\* [\_num] | 10, AEXP

------------------+----------

+ [\_num] | 10, AEXP

------------------+----------

- [\_num] | 10, AEXP

------------------+----------

/ [\_num] | 10, AEXP

------------------+----------

# [\_num] | 10, AEXP

------------------+----------

( [\_rc] | 11, AEXP

------------------+----------

\* [\_rc] | 11, AEXP

------------------+----------

+ [\_rc] | 11, AEXP

------------------+----------

- [\_rc] | 11, AEXP

------------------+----------

/ [\_rc] | 11, AEXP

------------------+----------

# [\_rc] | 11, AEXP  
 | ( | ) | \* | + | - | / | \_num | \_rc | $

----------+---+---+---+---+---+---+------+-----+---

[(, C] | S | S | S | S | S | S | S | S |

----------+---+---+---+---+---+---+------+-----+---

[), C] | | R | R | R | R | R | | | R

----------+---+---+---+---+---+---+------+-----+---

[\*, C] | S | R | R | R | R | R | S | S | R

----------+---+---+---+---+---+---+------+-----+---

[+, C] | S | R | S | R | R | S | S | S | R

----------+---+---+---+---+---+---+------+-----+---

[-, C] | S | R | S | R | R | S | S | S | R

----------+---+---+---+---+---+---+------+-----+---

[/, C] | S | R | R | R | R | R | S | S | R

----------+---+---+---+---+---+---+------+-----+---

[\_num, C] | | R | R | R | R | R | | | R

----------+---+---+---+---+---+---+------+-----+---

[\_rc, C] | | R | R | R | R | R | | | R

----------+---+---+---+---+---+---+------+-----+---

[#] | S | | S | S | S | S | S | S |

----------+---+---+---+---+---+---+------+-----+---

[#, AEXP] | S | | S | S | S | S | S | S | A

The relationship of precedence  
 | ( | ) | \* | + | - | / | \_num | \_rc | $

-----+------+------+------+------+------+------+------+------+------

( | [<.] | [=.] | [<.] | [<.] | [<.] | [<.] | [<.] | [<.] |

-----+------+------+------+------+------+------+------+------+------

) | | [.>] | [.>] | [.>] | [.>] | [.>] | | | [.>]

-----+------+------+------+------+------+------+------+------+------

\* | [<.] | [.>] | [.>] | [.>] | [.>] | [.>] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------+------

+ | [<.] | [.>] | [<.] | [.>] | [.>] | [<.] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------+------

- | [<.] | [.>] | [<.] | [.>] | [.>] | [<.] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------+------

/ | [<.] | [.>] | [.>] | [.>] | [.>] | [.>] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------+------

\_num | | [.>] | [.>] | [.>] | [.>] | [.>] | | | [.>]

-----+------+------+------+------+------+------+------+------+------

\_rc | | [.>] | [.>] | [.>] | [.>] | [.>] | | | [.>]

-----+------+------+------+------+------+------+------+------+------

# | [<.] | | [<.] | [<.] | [<.] | [<.] | [<.] | [<.] |

End.

File : "G8.TXT" .

NON-TERMINALS: [AEXP, ID, LEXP, LMULT, LTERM, RLTERM]

TERMINALS: [(, ), \_and, \_not, \_or, \_rel, \_str]

START SYMBOL: LEXP

RULES:

(0) LEXP -> LTERM

(1) LEXP -> LTERM \_or RLTERM

(2) RLTERM -> LTERM \_or RLTERM

(3) RLTERM -> LTERM

(4) LTERM -> LTERM \_and LMULT

(5) LTERM -> LMULT

(6) LMULT -> ID

(7) LMULT -> AEXP \_rel AEXP

(8) LMULT -> \_str \_rel \_str

(9) LMULT -> \_not LMULT

(10) LMULT -> ( LEXP )

The grammar of operator precedence

Table of transfer and convolution

| €

------------------------+----------

( [LEXP, \_or, LEXP] | 1, LEXP

------------------------+----------

\_or [LEXP, \_or, LEXP] | 1, LEXP

------------------------+----------

# [LEXP, \_or, LEXP] | 1, LEXP

------------------------+----------

( [LEXP, \_and, LEXP] | 4, LEXP

------------------------+----------

\_or [LEXP, \_and, LEXP] | 4, LEXP

------------------------+----------

# [LEXP, \_and, LEXP] | 4, LEXP

------------------------+----------

( [LEXP, \_rel, LEXP] | 7, LEXP

------------------------+----------

\_and [LEXP, \_rel, LEXP] | 7, LEXP

------------------------+----------

\_not [LEXP, \_rel, LEXP] | 7, LEXP

------------------------+----------

\_or [LEXP, \_rel, LEXP] | 7, LEXP

------------------------+----------

# [LEXP, \_rel, LEXP] | 7, LEXP

------------------------+----------

( [\_str, \_rel, \_str] | 8, LEXP

------------------------+----------

\_and [\_str, \_rel, \_str] | 8, LEXP

------------------------+----------

\_not [\_str, \_rel, \_str] | 8, LEXP

------------------------+----------

\_or [\_str, \_rel, \_str] | 8, LEXP

------------------------+----------

# [\_str, \_rel, \_str] | 8, LEXP

------------------------+----------

( [\_not, LEXP] | 9, LEXP

------------------------+----------

\_and [\_not, LEXP] | 9, LEXP

------------------------+----------

\_not [\_not, LEXP] | 9, LEXP

------------------------+----------

\_or [\_not, LEXP] | 9, LEXP

------------------------+----------

# [\_not, LEXP] | 9, LEXP

------------------------+----------

( [(, LEXP, )] | 10, LEXP

------------------------+----------

\_and [(, LEXP, )] | 10, LEXP

------------------------+----------

\_not [(, LEXP, )] | 10, LEXP

------------------------+----------

\_or [(, LEXP, )] | 10, LEXP

------------------------+----------

# [(, LEXP, )] | 10, LEXP

| ( | ) | \_and | \_not | \_or | \_rel | \_str | $

----------+---+---+------+------+-----+------+------+---

[(, C] | S | S | S | S | S | S | S |

----------+---+---+------+------+-----+------+------+---

[), C] | | R | R | | R | | | R

----------+---+---+------+------+-----+------+------+---

[\_and, C] | S | R | R | S | R | S | S | R

----------+---+---+------+------+-----+------+------+---

[\_not, C] | S | R | R | S | R | S | S | R

----------+---+---+------+------+-----+------+------+---

[\_or, C] | S | R | S | S | S | S | S | R

----------+---+---+------+------+-----+------+------+---

[\_rel, C] | | R | R | | R | | S | R

----------+---+---+------+------+-----+------+------+---

[\_str, C] | | R | R | | R | S | | R

----------+---+---+------+------+-----+------+------+---

[#] | S | | S | S | S | S | S |

----------+---+---+------+------+-----+------+------+---

[#, LEXP] | S | | S | S | S | S | S | A

The relationship of precedence

| ( | ) | \_and | \_not | \_or | \_rel | \_str | $

-----+------+------+------+------+------+------+------+------

( | [<.] | [=.] | [<.] | [<.] | [<.] | [<.] | [<.] |

-----+------+------+------+------+------+------+------+------

) | | [.>] | [.>] | | [.>] | | | [.>]

-----+------+------+------+------+------+------+------+------

\_and | [<.] | [.>] | [.>] | [<.] | [.>] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------

\_not | [<.] | [.>] | [.>] | [<.] | [.>] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------

\_or | [<.] | [.>] | [<.] | [<.] | [<.] | [<.] | [<.] | [.>]

-----+------+------+------+------+------+------+------+------

\_rel | | [.>] | [.>] | | [.>] | | [=.] | [.>]

-----+------+------+------+------+------+------+------+------

\_str | | [.>] | [.>] | | [.>] | [=.] | | [.>]

-----+------+------+------+------+------+------+------+------

# | [<.] | | [<.] | [<.] | [<.] | [<.] | [<.] |

End.

File : "G9.TXT" .

NON-TERMINALS: [BC, C, DEC]

TERMINALS: [+, -, ., \_cc, \_false, \_num, \_rc, \_str, \_true]

START SYMBOL: C

RULES:

(0) C -> DEC

(1) C -> \_cc

(2) C -> BC

(3) C -> \_rc

(4) DEC -> + \_num

(5) DEC -> - \_num

(6) DEC -> \_num

(7) BC -> \_true

(8) BC -> \_false

(9) C -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

------------+------

# [\_cc] | 1, C

------------+------

# [\_rc] | 3, C

------------+------

# [+, \_num] | 4, C

------------+------

# [-, \_num] | 5, C

------------+------

# [\_num] | 6, C

------------+------

# [\_true] | 7, C

------------+------

# [\_false] | 8, C

------------+------

# [\_str] | 9, C

| + | - | . | \_cc | \_false | \_num | \_rc | \_str | \_true | $

------------+---+---+---+-----+--------+------+-----+------+-------+---

[+, C] | | | | | | S | | | |

------------+---+---+---+-----+--------+------+-----+------+-------+---

[-, C] | | | | | | S | | | |

------------+---+---+---+-----+--------+------+-----+------+-------+---

[., C] | | | | | | | | | |

------------+---+---+---+-----+--------+------+-----+------+-------+---

[\_cc, C] | | | | | | | | | | R

------------+---+---+---+-----+--------+------+-----+------+-------+---

[\_false, C] | | | | | | | | | | R

------------+---+---+---+-----+--------+------+-----+------+-------+---

[\_num, C] | | | | | | | | | | R

------------+---+---+---+-----+--------+------+-----+------+-------+---

[\_rc, C] | | | | | | | | | | R

------------+---+---+---+-----+--------+------+-----+------+-------+---

[\_str, C] | | | | | | | | | | R

------------+---+---+---+-----+--------+------+-----+------+-------+---

[\_true, C] | | | | | | | | | | R

------------+---+---+---+-----+--------+------+-----+------+-------+---

[#] | S | S | | S | S | S | S | S | S |

------------+---+---+---+-----+--------+------+-----+------+-------+---

[#, C] | S | S | | S | S | S | S | S | S | A

The relationship of precedence

| + | - | . | \_cc | \_false | \_num | \_rc | \_str | \_true | $

-------+------+------+---+------+--------+------+------+------+-------+------

+ | | | | | | [=.] | | | |

-------+------+------+---+------+--------+------+------+------+-------+------

- | | | | | | [=.] | | | |

-------+------+------+---+------+--------+------+------+------+-------+------

. | | | | | | | | | |

-------+------+------+---+------+--------+------+------+------+-------+------

\_cc | | | | | | | | | | [.>]

-------+------+------+---+------+--------+------+------+------+-------+------

\_false | | | | | | | | | | [.>]

-------+------+------+---+------+--------+------+------+------+-------+------

\_num | | | | | | | | | | [.>]

-------+------+------+---+------+--------+------+------+------+-------+------

\_rc | | | | | | | | | | [.>]

-------+------+------+---+------+--------+------+------+------+-------+------

\_str | | | | | | | | | | [.>]

-------+------+------+---+------+--------+------+------+------+-------+------

\_true | | | | | | | | | | [.>]

-------+------+------+---+------+--------+------+------+------+-------+------

# | [<.] | [<.] | | [<.] | [<.] | [<.] | [<.] | [<.] | [<.] |

End.

File : "G10.TXT" .

NON-TERMINALS: [AOP, COP, FOROP, GOP, ID, IFAOP, IFOP, NEWOP, OP, OPOP, RECUROP, ROP, UMOP, WOP]  
TERMINALS: [:]  
START SYMBOL: OP  
 RULES:  
(0) OP -> ID : UMOP

(1) OP -> UMOP

(2) UMOP -> AOP

(3) UMOP -> GOP

(4) UMOP -> WOP

(5) UMOP -> ROP

(6) UMOP -> COP

(7) UMOP -> IFOP

(8) UMOP -> FOROP

(9) UMOP -> NEWOP

(10) UMOP -> IFAOP  
The grammar of operator precedence

Table of transfer and convolution  
 | €

--------------+-------

# [OP, :, OP] | 0, OP

| : | $

--------+---+---

[:, C] | | R

--------+---+---

[#] | S |

--------+---+---

[#, OP] | S | A

The relationship of precedence  
 | : | $

--+------+------

: | | [.>]

--+------+------

# | [<.] |

End.

File : "G11.TXT" .

NON-TERMINALS: [AEXP, ALEXP, AOP, C, CONOP, ID, LEXP, REPOP, SUBOP]  
TERMINALS: [\_eq]  
START SYMBOL: AOP  
 RULES:  
(0) AOP -> ID \_eq ALEXP  
(1) ALEXP -> C  
(2) ALEXP -> AEXP  
(3) ALEXP -> LEXP  
(4) ALEXP -> CONOP  
(5) ALEXP -> REPOP  
(6) ALEXP -> SUBOP  
The grammar of operator precedence

Table of transfer and convolution  
 | €   
------------------+--------  
# [AOP, \_eq, AOP] | 0, AOP  
  
 | \_eq | $  
---------+-----+---  
[\_eq, C] | | R  
---------+-----+---  
 [#] | S |   
---------+-----+---  
[#, AOP] | S | A  
  
The relationship of precedence  
 | \_eq | $   
----+------+------  
\_eq | | [.>]  
----+------+------  
 # | [<.] |

End.

File : "G12.TXT" .

NON-TERMINALS: [GOP, ID]

TERMINALS: [\_goto]

START SYMBOL: GOP

RULES:

(0) GOP -> \_goto ID

The grammar of operator precedence

Table of transfer and convolution

| €

---------------+--------

# [\_goto, GOP] | 0, GOP

| \_goto | $

-----------+-------+---

[\_goto, C] | | R

-----------+-------+---

[#] | S |

-----------+-------+---

[#, GOP] | S | A

The relationship of precedence

| \_goto | $

------+-------+------

\_goto | | [.>]

------+-------+------

# | [<.] |

End.

File : "G13.TXT" .

NON-TERMINALS: [AEXP, LEXP, MSG, RMSG, WOP]

TERMINALS: [(, ), ,, \_str, \_write]

START SYMBOL: WOP

RULES:

(0) WOP -> \_write ( MSG )

(1) RMSG -> MSG , RMSG

(2) RMSG -> MSG

(3) MSG -> AEXP

(4) MSG -> LEXP

(5) MSG -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

----------------------+--------

# [\_write, (, WOP, )] | 0, WOP

----------------------+--------

( [WOP, ,, WOP] | 1, WOP

----------------------+--------

, [WOP, ,, WOP] | 1, WOP

----------------------+--------

( [\_str] | 5, WOP

----------------------+--------

, [\_str] | 5, WOP

| ( | ) | , | \_str | \_write | $

------------+---+---+---+------+--------+---

[(, C] | | S | S | S | |

------------+---+---+---+------+--------+---

[), C] | | | | | | R

------------+---+---+---+------+--------+---

[,, C] | | R | S | S | |

------------+---+---+---+------+--------+---

[\_str, C] | | R | R | | |

------------+---+---+---+------+--------+---

[\_write, C] | S | | | | |

------------+---+---+---+------+--------+---

[#] | | | | | S |

------------+---+---+---+------+--------+---

[#, WOP] | | | | | S | A

The relationship of precedence

| ( | ) | , | \_str | \_write | $

-------+------+------+------+------+--------+------

( | | [=.] | [<.] | [<.] | |

-------+------+------+------+------+--------+------

) | | | | | | [.>]

-------+------+------+------+------+--------+------

, | | [.>] | [<.] | [<.] | |

-------+------+------+------+------+--------+------

\_str | | [.>] | [.>] | | |

-------+------+------+------+------+--------+------

\_write | [=.] | | | | |

-------+------+------+------+------+--------+------

# | | | | | [<.] |

End.

File : "G14.TXT" .

NON-TERMINALS: [ID, RID, ROP]

TERMINALS: [(, ), ,, \_read]

START SYMBOL: ROP

RULES:

(0) ROP -> \_read ( RID )

(1) RID -> ID , RID

(2) RID -> ID

The grammar of operator precedence

Table of transfer and convolution

| €

---------------------+--------

# [\_read, (, ROP, )] | 0, ROP

---------------------+--------

( [ROP, ,, ROP] | 1, ROP

---------------------+--------

, [ROP, ,, ROP] | 1, ROP

| ( | ) | , | \_read | $

-----------+---+---+---+-------+---

[(, C] | | S | S | |

-----------+---+---+---+-------+---

[), C] | | | | | R

-----------+---+---+---+-------+---

[,, C] | | R | S | |

-----------+---+---+---+-------+---

[\_read, C] | S | | | |

-----------+---+---+---+-------+---

[#] | | | | S |

-----------+---+---+---+-------+---

[#, ROP] | | | | S | A

The relationship of precedence

| ( | ) | , | \_read | $

------+------+------+------+-------+------

( | | [=.] | [<.] | |

------+------+------+------+-------+------

) | | | | | [.>]

------+------+------+------+-------+------

, | | [.>] | [<.] | |

------+------+------+------+-------+------

\_read | [=.] | | | |

------+------+------+------+-------+------

# | | | | [<.] |

End.

File : "G15.TXT" .

NON-TERMINALS: [COP, OP, ZOMOP]

TERMINALS: [;, \_begin, \_end]

START SYMBOL: COP

RULES:

(0) COP -> \_begin ZOMOP \_end

(1) COP -> \_begin \_end

(2) ZOMOP -> OP ; ZOMOP

(3) ZOMOP -> OP ;

The grammar of operator precedence

Table of transfer and convolution

| €

----------------------+--------

# [\_begin, COP, \_end] | 0, COP

----------------------+--------

# [\_begin, \_end] | 1, COP

----------------------+--------

; [COP, ;, COP] | 2, COP

----------------------+--------

\_begin [COP, ;, COP] | 2, COP

----------------------+--------

; [COP, ;] | 3, COP

----------------------+--------

\_begin [COP, ;] | 3, COP

| ; | \_begin | \_end | $

------------+---+--------+------+---

[;, C] | S | | R |

------------+---+--------+------+---

[\_begin, C] | S | | S |

------------+---+--------+------+---

[\_end, C] | | | | R

------------+---+--------+------+---

[#] | | S | |

------------+---+--------+------+---

[#, COP] | | S | | A

The relationship of precedence

| ; | \_begin | \_end | $

-------+------+--------+------+------

; | [<.] | | [.>] |

-------+------+--------+------+------

\_begin | [<.] | | [=.] |

-------+------+--------+------+------

\_end | | | | [.>]

-------+------+--------+------+------

# | | [<.] | |

End.

File : "G16.TXT" .

NON-TERMINALS: [COP, IFOP, LEXP, OCOP, OP]

TERMINALS: [\_else, \_if, \_then]

START SYMBOL: IFOP

RULES:

(0) IFOP -> \_if LEXP \_then OCOP

(1) IFOP -> \_if LEXP \_then OCOP \_else OCOP

(2) OCOP -> COP

(3) OCOP -> OP

The grammar of operator precedence

Table of transfer and convolution

| €

----------------------------------------+---------

# [\_if, IFOP, \_then, IFOP] | 0, IFOP

----------------------------------------+---------

# [\_if, IFOP, \_then, IFOP, \_else, IFOP] | 1, IFOP

| \_else | \_if | \_then | $

-----------+-------+-----+-------+---

[\_else, C] | | | | R

-----------+-------+-----+-------+---

[\_if, C] | | | S |

-----------+-------+-----+-------+---

[\_then, C] | S | | | R

-----------+-------+-----+-------+---

[#] | | S | |

-----------+-------+-----+-------+---

[#, IFOP] | | S | | A

The relationship of precedence

| \_else | \_if | \_then | $

------+-------+------+-------+------

\_else | | | | [.>]

------+-------+------+-------+------

\_if | | | [=.] |

------+-------+------+-------+------

\_then | [=.] | | | [.>]

------+-------+------+-------+------

# | | [<.] | |

End.

File : "G17.TXT" .

NON-TERMINALS: [COP, FOROP, LEXP, OCOP, OP]

TERMINALS: [\_do, \_while]

START SYMBOL: FOROP

RULES:

(0) FOROP -> \_while LEXP \_do OCOP

The grammar of operator precedence

Table of transfer and convolution

| €

------------------------------+----------

# [\_while, FOROP, \_do, FOROP] | 0, FOROP

| \_do | \_while | $

------------+-----+--------+---

[\_do, C] | | | R

------------+-----+--------+---

[\_while, C] | S | |

------------+-----+--------+---

[#] | | S |

------------+-----+--------+---

[#, FOROP] | | S | A

The relationship of precedence

| \_do | \_while | $

-------+------+--------+------

\_do | | | [.>]

-------+------+--------+------

\_while | [=.] | |

-------+------+--------+------

# | | [<.] |

End.

File : "G18.TXT" .

NON-TERMINALS: [ID, IDSC, LENOP]

TERMINALS: [(, ), \_len, \_str]

START SYMBOL: LENOP

RULES:

(0) LENOP -> \_len ( IDSC )

(1) IDSC -> ID

(2) IDSC -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

----------------------+----------

# [\_len, (, LENOP, )] | 0, LENOP

----------------------+----------

( [\_str] | 2, LENOP

| ( | ) | \_len | \_str | $

-----------+---+---+------+------+---

[(, C] | | S | | S |

-----------+---+---+------+------+---

[), C] | | | | | R

-----------+---+---+------+------+---

[\_len, C] | S | | | |

-----------+---+---+------+------+---

[\_str, C] | | R | | |

-----------+---+---+------+------+---

[#] | | | S | |

-----------+---+---+------+------+---

[#, LENOP] | | | S | | A

The relationship of precedence

| ( | ) | \_len | \_str | $

-----+------+------+------+------+------

( | | [=.] | | [<.] |

-----+------+------+------+------+------

) | | | | | [.>]

-----+------+------+------+------+------

\_len | [=.] | | | |

-----+------+------+------+------+------

\_str | | [.>] | | |

-----+------+------+------+------+------

# | | | [<.] | |

End.

File : "G19.TXT" .

NON-TERMINALS: [CONOP, ID, IDSC]

TERMINALS: [(, ), ,, \_conc, \_str]

START SYMBOL: CONOP

RULES:

1. CONOP -> \_conc ( IDSC , IDSC )
2. IDSC -> ID
3. IDSC -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

---------------------------------+----------

# [\_conc, (, CONOP, ,, CONOP, )] | 0, CONOP

---------------------------------+----------

( [\_str] | 2, CONOP

---------------------------------+----------

, [\_str] | 2, CONOP

| ( | ) | , | \_conc | \_str | $

-----------+---+---+---+-------+------+---

[(, C] | | | S | | S |

-----------+---+---+---+-------+------+---

[), C] | | | | | | R

-----------+---+---+---+-------+------+---

[,, C] | | S | | | S |

-----------+---+---+---+-------+------+---

[\_conc, C] | S | | | | |

-----------+---+---+---+-------+------+---

[\_str, C] | | R | R | | |

-----------+---+---+---+-------+------+---

[#] | | | | S | |

-----------+---+---+---+-------+------+---

[#, CONOP] | | | | S | | A

The relationship of precedence

| ( | ) | , | \_conc | \_str | $

------+------+------+------+-------+------+------

( | | | [=.] | | [<.] |

------+------+------+------+-------+------+------

) | | | | | | [.>]

------+------+------+------+-------+------+------

, | | [=.] | | | [<.] |

------+------+------+------+-------+------+------

\_conc | [=.] | | | | |

------+------+------+------+-------+------+------

\_str | | [.>] | [.>] | | |

------+------+------+------+-------+------+------

# | | | | [<.] | |

End.

File : "G20.TXT" .

NON-TERMINALS: [ID, IDSC, REPOP]

TERMINALS: [(, ), ,, \_repl, \_str]

START SYMBOL: REPOP

RULES:

(0) REPOP -> \_repl ( IDSC , IDSC , IDSC )

(1) IDSC -> ID

(2) IDSC -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

-------------------------------------------+----------

# [\_repl, (, REPOP, ,, REPOP, ,, REPOP, )] | 0, REPOP

-------------------------------------------+----------

( [\_str] | 2, REPOP

-------------------------------------------+----------

, [\_str] | 2, REPOP

| ( | ) | , | \_repl | \_str | $

-----------+---+---+---+-------+------+---

[(, C] | | | S | | S |

-----------+---+---+---+-------+------+---

[), C] | | | | | | R

-----------+---+---+---+-------+------+---

[,, C] | | S | S | | S |

-----------+---+---+---+-------+------+---

[\_repl, C] | S | | | | |

-----------+---+---+---+-------+------+---

[\_str, C] | | R | R | | |

-----------+---+---+---+-------+------+---

[#] | | | | S | |

-----------+---+---+---+-------+------+---

[#, REPOP] | | | | S | | A

The relationship of precedence

| ( | ) | , | \_repl | \_str | $

------+------+------+------+-------+------+------

( | | | [=.] | | [<.] |

------+------+------+------+-------+------+------

) | | | | | | [.>]

------+------+------+------+-------+------+------

, | | [=.] | [=.] | | [<.] |

------+------+------+------+-------+------+------

\_repl | [=.] | | | | |

------+------+------+------+-------+------+------

\_str | | [.>] | [.>] | | |

------+------+------+------+-------+------+------

# | | | | [<.] | |

End.

File : "G21.TXT" .

NON-TERMINALS: [ID, IDSC, SEOP]

TERMINALS: [(, ), ,, \_srch, \_str]

START SYMBOL: SEOP

RULES:

(0) SEOP -> \_srch ( IDSC , IDSC )

(1) IDSC -> ID

(2) IDSC -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

-------------------------------+---------

# [\_srch, (, SEOP, ,, SEOP, )] | 0, SEOP

-------------------------------+---------

( [\_str] | 2, SEOP

-------------------------------+---------

, [\_str] | 2, SEOP

| ( | ) | , | \_srch | \_str | $

-----------+---+---+---+-------+------+---

[(, C] | | | S | | S |

-----------+---+---+---+-------+------+---

[), C] | | | | | | R

-----------+---+---+---+-------+------+---

[,, C] | | S | | | S |

-----------+---+---+---+-------+------+---

[\_srch, C] | S | | | | |

-----------+---+---+---+-------+------+---

[\_str, C] | | R | R | | |

-----------+---+---+---+-------+------+---

[#] | | | | S | |

-----------+---+---+---+-------+------+---

[#, SEOP] | | | | S | | A

The relationship of precedence

| ( | ) | , | \_srch | \_str | $

------+------+------+------+-------+------+------

( | | | [=.] | | [<.] |

------+------+------+------+-------+------+------

) | | | | | | [.>]

------+------+------+------+-------+------+------

, | | [=.] | | | [<.] |

------+------+------+------+-------+------+------

\_srch | [=.] | | | | |

------+------+------+------+-------+------+------

\_str | | [.>] | [.>] | | |

------+------+------+------+-------+------+------

# | | | | [<.] | |

End.

File : "G22.TXT" .

NON-TERMINALS: [ID, IDNN, IDSC, SUBOP]

TERMINALS: [(, ), ,, \_num, \_str, \_subs]

START SYMBOL: SUBOP

RULES:

(0) SUBOP -> \_subs ( IDSC , IDNN , IDNN )

(1) IDNN -> ID

(2) IDNN -> \_num

(3) IDSC -> ID

(4) IDSC -> \_str

The grammar of operator precedence

Table of transfer and convolution

| €

-------------------------------------------+----------

# [\_subs, (, SUBOP, ,, SUBOP, ,, SUBOP, )] | 0, SUBOP

-------------------------------------------+----------

, [\_num] | 2, SUBOP

-------------------------------------------+----------

( [\_str] | 4, SUBOP

| ( | ) | , | \_num | \_str | \_subs | $

-----------+---+---+---+------+------+-------+---

[(, C] | | | S | | S | |

-----------+---+---+---+------+------+-------+---

[), C] | | | | | | | R

-----------+---+---+---+------+------+-------+---

[,, C] | | S | S | S | | |

-----------+---+---+---+------+------+-------+---

[\_num, C] | | R | R | | | |

-----------+---+---+---+------+------+-------+---

[\_str, C] | | | R | | | |

-----------+---+---+---+------+------+-------+---

[\_subs, C] | S | | | | | |

-----------+---+---+---+------+------+-------+---

[#] | | | | | | S |

-----------+---+---+---+------+------+-------+---

[#, SUBOP] | | | | | | S | A

The relationship of precedence

| ( | ) | , | \_num | \_str | \_subs | $

------+------+------+------+------+------+-------+------

( | | | [=.] | | [<.] | |

------+------+------+------+------+------+-------+------

) | | | | | | | [.>]

------+------+------+------+------+------+-------+------

, | | [=.] | [=.] | [<.] | | |

------+------+------+------+------+------+-------+------

\_num | | [.>] | [.>] | | | |

------+------+------+------+------+------+-------+------

\_str | | | [.>] | | | |

------+------+------+------+------+------+-------+------

\_subs | [=.] | | | | | |

------+------+------+------+------+------+-------+------

# | | | | | | [<.] |

End.

File : "G23.TXT" .

NON-TERMINALS: [CLV, CONSTS, FUNCS, LABELS, VARS]

TERMINALS: [;]

START SYMBOL: CLV

RULES:

(0) CLV -> CONSTS

(1) CLV -> CONSTS ; CLV

(2) CLV -> LABELS

(3) CLV -> LABELS ; CLV

(4) CLV -> VARS

(5) CLV -> VARS ; CLV

(6) CLV -> FUNCS

(7) CLV -> FUNCS ; CLV

The grammar of operator precedence

Table of transfer and convolution

| €

----------------+--------

; [CLV, ;, CLV] | 1, CLV

----------------+--------

# [CLV, ;, CLV] | 1, CLV

| ; | $

---------+---+---

[;, C] | S | R

---------+---+---

[#] | S |

---------+---+---

[#, CLV] | S | A

The relationship of precedence

| ; | $

--+------+------

; | [<.] | [.>]

--+------+------

# | [<.] |

End.

File : "G25.TXT" .

NON-TERMINALS: [AOP, IFAOP, LEXP, OEAOP]

TERMINALS: [\_else, \_if, \_then]

START SYMBOL: IFAOP

RULES:

(0) IFAOP -> \_if LEXP \_then AOP

(1) IFAOP -> \_if LEXP \_then AOP \_else AOP

The grammar of operator precedence

Table of transfer and convolution

| €

-------------------------------------------+----------

# [\_if, IFAOP, \_then, IFAOP] | 0, IFAOP

-------------------------------------------+----------

# [\_if, IFAOP, \_then, IFAOP, \_else, IFAOP] | 1, IFAOP

| \_else | \_if | \_then | $

-----------+-------+-----+-------+---

[\_else, C] | | | | R

-----------+-------+-----+-------+---

[\_if, C] | | | S |

-----------+-------+-----+-------+---

[\_then, C] | S | | | R

-----------+-------+-----+-------+---

[#] | | S | |

-----------+-------+-----+-------+---

[#, IFAOP] | | S | | A

The relationship of precedence

| \_else | \_if | \_then | $

------+-------+------+-------+------

\_else | | | | [.>]

------+-------+------+-------+------

\_if | | | [=.] |

------+-------+------+-------+------

\_then | [=.] | | | [.>]

------+-------+------+-------+------

# | | [<.] | |

End.

File : "G26.TXT" .

NON-TERMINALS: [COP, ID, OOP, OPOP, OVAR, OVAR1, OVAR2, TYPE]

TERMINALS: [(, ), \*, +, ,, -, /, :, ;, \_oper, \_rel]

START SYMBOL: OPOP

RULES:

(0) OPOP -> \_oper OOP ( OVAR ) ID : TYPE ; COP

(1) OVAR -> OVAR1

(2) OVAR -> OVAR2

(3) OVAR1 -> ID : TYPE ; ID : TYPE

(4) OVAR2 -> ID , ID : TYPE

(5) OOP -> \_rel

(6) OOP -> +

(7) OOP -> -

(8) OOP -> \*

(9) OOP -> /

The grammar of operator precedence

Table of transfer and convolution

| €

----------------------------------------------------+---------

# [\_oper, OPOP, (, OPOP, ), OPOP, :, OPOP, ;, OPOP] | 0, OPOP

----------------------------------------------------+---------

( [OPOP, :, OPOP, ;, OPOP, :, OPOP] | 3, OPOP

----------------------------------------------------+---------

( [OPOP, ,, OPOP, :, OPOP] | 4, OPOP

----------------------------------------------------+---------

\_oper [\_rel] | 5, OPOP

----------------------------------------------------+---------

\_oper [+] | 6, OPOP

----------------------------------------------------+---------

\_oper [-] | 7, OPOP

----------------------------------------------------+---------

\_oper [\*] | 8, OPOP

----------------------------------------------------+---------

\_oper [/] | 9, OPOP

| ( | ) | \* | + | , | - | / | : | ; | \_oper | \_rel | $

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[(, C] | | S | | | S | | | S | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[), C] | | | | | | | | S | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[\*, C] | R | | | | | | | | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[+, C] | R | | | | | | | | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[,, C] | | | | | | | | S | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[-, C] | R | | | | | | | | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[/, C] | R | | | | | | | | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[:, C] | | R | | | | | | | S | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[;, C] | | | | | | | | S | | | | R

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[\_oper, C] | S | | S | S | | S | S | | | | S |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[\_rel, C] | R | | | | | | | | | | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[#] | | | | | | | | | | S | |

-----------+---+---+---+---+---+---+---+---+---+-------+------+---

[#, OPOP] | | | | | | | | | | S | | A

The relationship of precedence

| ( | ) | \* | + | , | - | / | : | ; | \_oper | \_rel | $

------+------+------+------+------+------+------+------+------+------+-------+------+------

( | | [=.] | | | [<.] | | | [<.] | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

) | | | | | | | | [=.] | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

\* | [.>] | | | | | | | | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

+ | [.>] | | | | | | | | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

, | | | | | | | | [=.] | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

- | [.>] | | | | | | | | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

/ | [.>] | | | | | | | | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

: | | [.>] | | | | | | | [=.] | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

; | | | | | | | | [=.] | | | | [.>]

------+------+------+------+------+------+------+------+------+------+-------+------+------

\_oper | [=.] | | [<.] | [<.] | | [<.] | [<.] | | | | [<.] |

------+------+------+------+------+------+------+------+------+------+-------+------+------

\_rel | [.>] | | | | | | | | | | |

------+------+------+------+------+------+------+------+------+------+-------+------+------

# | | | | | | | | | | [<.] | |

End.

## Analysis of the chain DMP-converter.

program HelloWorld;

var a:integer; b:boolean;

begin

a:=3;

write(‘hello world’);

end.

Formed symbol table:

Table variables:

|  |  |  |  |
| --- | --- | --- | --- |
| **Offset** | **Variable identifier** | **Variable type** | **The value of the variable** |
| 1 | "a" | integer | 3 |
| 2 | “b” | boolean | true |

Table tags and constants have remained empty because the program has never encountered this type of ads identifiers.

Chain:

\_PROG \_ID ; \_VAR \_ID : \_INT ; \_ID : \_BOOL ; \_BEGIN \_ID \_EQ \_NUM ; \_WRITE ( \_STR ) ; \_END .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Stack** | **Input chain** | **Output chain** | **Action** | |
| ***G0: ┴*** | *\_PROG \_ID ; \_VAR \_ID…* |  | Transfer | |
| ***G0: ┴ \_PROG*** | *\_ID ; \_VAR \_ID…* |  | Go to G6 | |
| ***G6: ┴*** | *\_ID ; \_VAR \_ID…* |  | Transfer | |
| ***G6: ┴ \_ID*** | *; \_VAR \_ID…* | ID → \_ID | Conv, 0 | |
| ***G6: ┴ ID*** | *; \_VAR \_ID…* |  | Admit (Go to G0) |
| ***G0: ┴ \_PROG ID*** | *; \_VAR \_ID…* |  | Transfer | |
| ***G0: ┴ \_PROG ID ;*** | *\_VAR \_ID : \_INT ;…* |  | Go to G1 | |
| ***G1: ┴*** | *\_VAR \_ID : \_INT ;…* |  | Transfer | |
| ***G1: ┴ ­\_VAR*** | *\_ID : \_INT ;…* |  | Go to G6 | |
| ***G6: ┴ ­*** | *\_ID : \_INT ; \_ID : \_BOOL ;* |  | Transfer |
| ***G6: ┴ ­\_ID*** | *: \_INT ; \_ID : \_BOOL ; \_BEGIN…* | ID → \_ID | Conv, 0 |
| ***G6: ┴ ­ID*** | *: \_INT ; \_ID : \_BOOL ; \_BEGIN…* |  | Admit (Go to G1) | |
| ***G1: ┴ ­\_VAR ID*** | *: \_INT ; \_ID : \_BOOL ; \_BEGIN…* |  | Transfer | |
| ***G1: ┴ ­\_VAR ID :*** | *\_INT ; \_ID : \_BOOL ; \_BEGIN…* |  | Go to G5 | |
| ***G5: ┴*** | *\_INT ; \_ID : \_BOOL ; \_BEGIN…* |  | Transfer | |
| ***G5: ┴ ­\_INT*** | *; \_ID : \_BOOL ; \_BEGIN…* | TYPE → \_INT | Conv, 0 |
| ***G5: ┴ ­TYPE*** | *; \_ID : \_BOOL ; \_BEGIN…* |  | Admit (Go to G1) | |
| ***G1: ┴ ­\_VAR ID : TYPE*** | *; \_ID : \_BOOL ; \_BEGIN…* | VAR → ID : TYPE | Conv, 3 | |
| ***G1: ┴ ­\_VAR VAR*** | *; \_ID : \_BOOL ; \_BEGIN…* |  | Transfer | |
| ***G1: ┴ ­\_VAR VAR ;*** | *\_ID : \_BOOL ; \_BEGIN…* |  | Go to G6 | |
| ***G6: ┴ ­*** | *\_ID : \_BOOL ; \_BEGIN…* |  | Transfer | |
| ***G6: ┴ ­ \_ID*** | *: \_BOOL ; \_BEGIN…* | ID → \_ID | Conv, 0 |
| ***G6: ┴ ­ ID*** | *: \_BOOL ; \_BEGIN…* |  | Admit (Go to G1) | |
| ***G1: ┴ ­\_VAR VAR ; ID*** | *: \_BOOL ; \_BEGIN…* |  | Transfer | |
| ***G1: ┴ ­\_VAR VAR ; ID :*** | *\_BOOL ; \_BEGIN…* |  | Go to G5 | |
| ***G5: ┴ ­*** | *\_BOOL ; \_BEGIN…* |  | Transfer | |
| ***G5: ┴ ­\_BOOL*** | *; \_BEGIN…* | TYPE → \_BOOL | Conv, 0 | |
| ***G5: ┴ ­TYPE*** | *; \_BEGIN…* |  | Admit (Go to G1) | |
| ***G1: ┴ ­\_VAR VAR ; ID : TYPE*** | *; \_BEGIN…* | VAR → ID : TYPE | Conv, 3 | |
| ***G1: ┴ ­\_VAR VAR ; VAR*** | *; \_BEGIN…* | RVAR → VAR ; RVAR | Conv, 1 | |
| ***G1: ┴ ­\_VAR RVAR*** | *; \_BEGIN …* |  | Conv, 0 | |
| ***G1: ┴ ­VARS*** | *; \_BEGIN …* |  | Admit (Go to G0) |
| ***G0: ┴ \_PROG ID ; VARS*** | *; \_BEGIN \_ID \_EQ \_NUM ;…* |  | Transfer |
| ***G0: ┴ \_PROG ID ; VARS ;*** | *\_BEGIN \_ID \_EQ \_NUM ;…* |  | Go to G15 | |
| ***G15: ┴ ­*** | *\_BEGIN \_ID \_EQ \_NUM ;…* |  | Transfer | |
| ***G15: ┴ ­\_BEGIN*** | *\_ID \_EQ \_NUM ;…* |  | Go to G6 | |
| ***G6: ┴ ­*** | *\_ID \_EQ \_NUM ;…* |  | Transfer |
| ***G6: ┴ ­ \_ID*** | *\_EQ \_NUM ;…* | ID → \_ID | Conv, 0 | |
| ***G6: ┴ ­ ID*** | *\_EQ \_NUM ;…* |  | Admit (Go to G15) | |
| ***G15: ┴ ­\_BEGIN ID*** | *\_EQ \_NUM ; \_WRITE ( \_STR ) ;…* |  | Go to G11 | |
| ***G11: ┴ ­*** | *\_EQ \_NUM ; \_WRITE ( \_STR ) ;…* |  | Transfer | |
| ***G11: ┴ ­\_EQ*** | *\_NUM ; \_WRITE ( \_STR ) ;…* |  | Go to G9 |
| ***G9: ┴ ­*** | *\_NUM ; \_WRITE ( \_STR ) ;…* |  | Transfer | |
| ***G9: ┴ ­\_NUM*** | *; \_WRITE ( \_STR ) ;…* | C → DEC → \_NUM | Conv, 7, 0 | |
| ***G9: ┴ ­C*** | *; \_WRITE ( \_STR ) ;…* |  | Admit (Go to G11) | |
| ***G11: ┴ ­\_EQ C*** | *; \_WRITE ( \_STR ) ;…* | AOP → ID \_EQ ALEXP | Conv, 0 | |
| ***G11: ┴ ­AOP*** | *; \_WRITE ( \_STR ) ;…* |  | Admit (Go to G15) |
| ***G15: ┴ ­\_BEGIN AOP*** | *; \_WRITE ( \_STR ) ; \_END .* |  | Transfer | |
| ***G15: ┴ ­\_BEGIN AOP ;*** | *\_WRITE ( \_STR ) ; \_END .* |  | Go to G13 | |
| ***G13: ┴*** | *\_WRITE ( \_STR ) ; \_END .* |  | Transfer | |
| ***G13: ┴ \_WRITE*** | *( \_STR ) ; \_END .* |  | Transfer | |
| ***G13: ┴ \_WRITE (*** | *\_STR ) ; \_END .* |  | Go to G9 | |
| ***G9: ┴*** | *\_STR ) ; \_END .* |  | Transfer |
| ***G9: ┴ \_STR*** | *) ; \_END .* | SC → \_STR | Conv, 15 | |
| ***G9: ┴ SC*** | *) ; \_END .* |  | Admit (Go to G13) | |
| ***G13: ┴ \_WRITE ( SC*** | *) ; \_END .* |  | Transfer | |
| ***G13: ┴ \_WRITE ( SC )*** | *; \_END .* | WOP → \_WRITE ( MSG ) | Conv, 0 | |
| ***G13: ┴ WOP*** | *; \_END .* |  | Admit (Go to G15) | |
| ***G15: ┴ ­\_BEGIN AOP ; WOP*** | *; \_END .* |  | Transfer | |
| ***G15: ┴ ­\_BEGIN AOP ; WOP ;*** | *\_END .* | ZOMOP → OP | Conv, 3 | |
| ***G15: ┴ ­\_BEGIN AOP ; ZOMOP*** | *\_END .* | ZOMOP → OP ; ZOMOP | Conv, 2 |
| ***G15: ┴ ­\_BEGIN ZOMOP*** | *\_END .* |  | Transfer | |
| ***G15: ┴ ­\_BEGIN ZOMOP END*** | *.* |  | Conv, 0 | |
| ***G15: ┴ ­COP*** | *.* |  | Admit (Go to G0) | |
| ***G0: ┴ \_PROG ID ; VARS ; COP*** | *.* |  | Transfer | |
| ***G0: ┴ \_PROG ID ; VARS ; COP .*** | *ε* | OCOP → COP . | Conv, 4 | |
| ***G0: ┴ \_PROG ID ; VARS ; OCOP*** | *ε* | OPROG → CLV ; OCOP | Conv, 3 | |
| ***G0: ┴ \_PROG ID ; OPROG*** | *ε* | PROG → \_PROG ID ; OPROG | Conv, 0 | |
| ***G0: ┴ PROG*** | *ε* |  | Admit | |

# Results

As a result of laboratory work was a description of the syntax and semantics of the language, to formalize the process of the syntax managed of translation and analyzed its example. It was implemented syntactical analysis.