Saint-Petersburg State Electro technical University “LETI”

**REPORT**

**Language processors development, laboratory work №1**

**«Lexical analyzer»**

Students: Stepanov K.S.

Ryabikov E.M.

Vasilyev D.A.

FKTI

Group: 9303

Variant 4

Hodirev I.A.

Saint-Petersbourg 2014

Assignment

Goal of this work is to develop a programming language, being set of a language, specified by individual task, and a translator from this language to an intermediate language of type, specified by individual task. Syntax analysis type is also mentioned in individual variant.

Language must support manipulations with variables and constants of specified types and also variables and components of derivative type, which is specified by individual task. Operation set must contain default language operations and operations, specified by individual task. Language must provide type conversion operation in case of types are equivalent. language must provide possibility of defining user types.

Language must provide arithmetic operations with constants, simple variables of default language types, structure type components, parentheses, signs (addition, substraction, division, multiplication) with usual operation priority.

Language must provide using logical expressions with relations, parentheses, and logical operations (and, or, not). If language contains logical constants they must be used too. Operation priority – usual.

Operations with structured type are specified by individual task.

Operators:

* assignment;
* input;
* output;
* complex;
* goto;
* conditional operator with logical expression as condition;
* cycle operator
* operators, specified by individual task

Also program should support comments.

**Задание 4**

*Base language* – Pascal.

*Base types:* Integer, rational, bounded, enumerable.

*Structured type* – Array.

*Rational type operations:* getting numerator and denominator, taking integer and fractional part, conversion to irregular fraction, calculating common denominator, simplification.

*Cycle operator* – with parameter.

*Operation overload* – permitted.

*Type equivalency* – by name.

*Grammar class* – LL(1).

*Intermediate representation*  – tetrad

# Language description

## Language syntax description.

<program>::=**program** <program\_name>;

[<section description>]

**begin**

[<section operators>]

**end.**

< program\_name >::=<id>

< section description >::=[**const** < const description >]

[**type** < types description >]

[**label** < label description >]

[**var** < var description >]< section description >

<описания\_типов>::=<имя\_пользовательского\_типа> = <тип>;

{< types description >}

<user type name>::=<id>

<type>::=<defined\_type>|< user type name >

<defined\_type>::=**Integer**|**Rational**|<array>|<bounded>|<enumerable>

<array>::=**array** [<id\_diap>{,<id\_diap>}] **of** <type>

< id\_diap >::=<unsigned int>..<unsigned int>

< user type name >::=<id>

<bounded>::=<signed int>..<signed int>

<enumerable>::=(<id>{,<id>})

<label description>::=<label>{,<label>};

<label>::=<id>

<const\_description>::=<const name> = <int const>;

{< const\_description >}

< int const >::=<signed int>|<const name>

< signed int >::=[+|-]<unsigned int>

< const name >::=<id>

<var description>::=<variable>{,< variable >} : <type>;

< variable >::=<id>

<operators description>::=<operator>;{< operator >;}

< operator >::=[<label>:]<unlabeled operator >

<unlabeled\_operator>::=<conditional\_operator>|<assignment\_operator>|<cycle\_operator>|<operator\_goto>|<input\_operator>|<output\_operator>|<complex\_operator>

< conditional\_operator >::= **if**<logical\_expression>**then**<operator> [**else**<operator>]

< logical\_expression >::=<disjunction\_operand>{**or**< disjunction\_operand >}

< disjunction\_operand >::=<conjunction\_operand>{**and**< conjunction\_operand >}

< disjunction\_operand >::=[**not**]<negation\_operang>

< negation\_operang >::=(<relation>)|(< logical\_expression >)

< relation >::=< relation operand>{< relation operand >< relation operand >}

< relation operand >::=<expression>

< relation operand >::= >|<|=|<>|<=|>=

<выражение>::=<signed integer>|(+|-)<vector element name>|(+|-)<variable>[.**numerator**|.**denominator**]|(+|-)<signed\_integer>|<operation>

<vector\_element\_name >::=<vactor\_name>[< vector\_element\_name >{,< vector\_element\_number >}]

<vector\_name>::=<id>

< vector\_element\_number >::=<unsigned int>

<operation>::=<operand\_addition>{(+|-) < operand\_addition >}

< operand\_addition >::=< operand\_multiplication>{(\*|/) < operand\_multiplication >}

<operand\_multiplication>::=<specific\_operator>(<operand\_multiplication>)|(<expression>)|<signed integer>|(+|-)< vector\_element\_number >| (+|-)<variable>[.**numerator**|.**denominator**]| (+|-)<integer constant>|<operator\_common\_denomenator>(<expression>,<expression>)

<specific operator>::=**int**|**frac**|**improper**|**simple**

<operator\_common\_denomenator>::=**comden**

<operator\_assignment>::=<variable>:=<expression>

<complex\_operator>::=**begin** <operator>;{<operator>;} **end**

<input\_operator>::=<input\_operator\_name>(<values>)

< input\_operator\_name >::=**read**|**readln**

<values>::=<variable>|<expression>

<output-operator>::=< output\_operator\_name >(<values>)

< output\_operator\_name >::=**write**|**writeln**

<goto\_operator>::=**goto**<label>  
<cycle\_operator>::=**for** <variable>:=<expression> **to** <expression> **do**

<operator>;

<id>::=<letter>|<sign\_underline> {<letter>|< sign\_underline >|<digit>}

<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

<unsigned int>::=<digit>{<digit>}

<digit> ::= 0 **|** 1 **|** 2 **|** 3 **|** 4 **|** 5 **|** 6 **|** 7 **|** 8 **|** 9

< sign\_underline >::= \_

## CS-Grammar

G = {N, T, R, S}

N ={ OPERATION , ODADD, ODM, SPECO, OPCOMDEN, LE, DISO, CONJO, NEGO, RATO, RATOD, VECN, ARRN, ARRNUMS, ARRNUM, V, SINT, L, OSET, O, CO, AO, FO, GO, INO, OUTO, COMO, , NINO, INM, NOUTO, OUTM, INITM, ENDM, T, ST, UT, ARR, IDIAPS, IDIAP, TDIAP, ENUM, IDSET, DV, VSET, VSETTALE, DL, LSET, LSETTALE, DC, NC, IC, DT, N, SD, SO

}

T ={ OPAR0, OPAR1, OPMUL0, OPMUL1, SPO0, SPC0, SCOM0, SP0, KNUM25, KDEN26, KINT27, KFRC28, KSIM30, KCD31, KOR18, KAND19, OPCP1, OPCP2, OPCP4, OPCP5, OPCP6, SBKO0, SBKC1, CINT , ID, SSCOL0, SCOL0, KIF7, KTH8, KEL9, OPA0, KBG5, KEN6, KRD21, KRDL22, KWT23, KWLT24, KGO13, KFR10, KTO11, KDO12 KTINT0, KTRAT1, KARR3, SBKC0, KOF4, OPINIT0 , KPR , KBG , KEN , KCON, KVAR, KTYP, KLAB}

R = {

S-> **KPR** N **SSCOL0** SD **KBG** SO **KEN** SP0

N-> **ID**

SD-> **KCON** DC SD

SD-> **KVAR** DV SD

SD-> **KTYP** DT SD

SD-> **KLAB** DL SD

SD-> ε

DT->UT **OPINIT0** T **SSCOL0** DT

DT-> ε

UT->**ID**

T->ST

T->UT

ST->**KTINT0**

ST->**KTRAT1**

ST->ARR

ST->TDIAP

ST->ENUM

ARR->**KARR3** **SBKO0** IDIAPS **SBKC0** **KOF4** T

IDIAPS->IDIAP **SCOM0** IDIAPS

IDIAPS->IDIAP

IDIAP->**CINT** **OPDPS0** **CINT**

TDIAP->SINT **OPDPS0** SINT

ENUM->**SPO0** IDSET **SPC0**

IDSET->**ID**

IDSET->**ID** **SCOM0** IDSET

DL->LSET **SSCOL0**

LSET->L

LSET->L **SCOM0** LSET

L->**ID**

DC->NC **OPINIT0** IC **SSCOL0** DC

IC->SINT

IC->**OPAR0** NC

IC->**OPAR1** NC

SINT->**OPAR0** **CINT**

SINT->**OPAR1** **CINT**

NC->**ID**

DV-> VSET **SCOL0** T **SSCOL0**

VSET-> V

VSET-> V **SCOM0** VSET

V->**ID**

SO->OSET

OSET->O **SSCOL0**

OSET->O **SSCOL0** OSET

O->L **SCOL0** UO

O->UO

UO->CO

UO->AO

UO->FO

UO->GO

UO->INO

UO->OUTO

UO->COMO

CO->**KIF7** LE **KTH8** O **KEL9** O

CO->**KIF7** LE **KTH8** O

LE->DISO

LE->DISO **KOR18** DISO

DISO->CONJO

DISO-> CONJO **KAND19** CONJO

CONJO-> **KNOT20** NEGO

CONJO->NEGO

NEGO->**SPO0** RATO **SPC0**

NEGO->**SPO0** LE **SPC0**

RATO->RATOD

RATO->RATOD RATOP RATOD

RATOD->EXP

RATOP->**OPCP1**

RATOP->**OPCP2**

RATOP->**OPCP4**

RATOP->**OPCP5**

RATOP->**OPCP6**

RATOP->**OPINIT0**

VECN->ARR **SBKO0** ARRNUMS **SBKC1**

ARRNUMS->ARRNUM

ARRNUMS->ARRNUM **SCOM0** ARRNUMS

ARRNUM->**CINT**

ARRN->**ID**

EXP->OPERATION

OPERATION->ODADD

OPERATION->ODADD **OPAR0** ODADD

OPERATION->ODADD **OPAR1** ODADD

ODADD->ODM

ODADD->ODM **OPMUL0** ODM

ODADD->ODM **OPMUL1** ODM

ODM->SPECO **SPO0** ODM **SPC0**

ODM->**SPO0** EXP **SPC0**

ODM->OPCOMDEN **SPO0** EXP **SCOM0** EXP **SPC0**

ODM->SINT

ODM->**OPAR0** VECN

ODM->**OPAR1** VECN

ODM->VECN

ODM->**OPAR0** V

ODM->**OPAR1** V

ODM->V

ODM->**OPAR0** V **SP0** **KNUM25**

ODM->**OPAR1** V **SP0** **KNUM25**

ODM->**OPAR0** V **SP0** **KDEN26**

ODM->**OPAR1** V **SP0** **KDEN26**

ODM->V **SP0** **KNUM25**

ODM->V SP0 **KDEN26**

SPECO->**KINT27**

SPECO->**KFRC28**

SPECO->**KIMP29**

SPECO->**KSIM30**

OPCOMDEN->**KCD31**

AO->V **OPA0** EXP

COMO->**KBG5** OSET **KEN6**

INO->NINO **SPO0** INM **SPC0**

NINO->**KRD21**

NINO->**KRDL22**

INM->V

OUTO->NOUTO **SPO0** OUTM **SPC0**

NOUTO->**KWT23**

NOUTO->**KWLT24**

OUTM->V

OUTM->EXP

GO->**KGO13** L

FO-> **KFR10** V **OPA0** INITM **KTO11** ENDM **KDO12** O

INITM->SINT

ENDM->SINT

}

## CS-Grammar transformed to LL1-type.

3.1 Grammar operation

GRAMMAR\_OPERATION={N,T,R,OPERATION}

N={ OPERATION , OPERAND\_ADD, OPERATIONTALE, OPERAND\_MUL, OPERAND\_ADDTALE, SPECO, OPCOMDEN, OPERAND\_MUL\_VTALE , OPERAND\_MUL\_TALE, OPERAND\_MUL\_VTALESP0, ARRNUMS, ARRNUM, ARRNUMTALE, OPERAND\_MUL\_UNS }

T={ OPAR0, OPAR1, OPMUL0, OPMUL1, SPO0, SPC0, SCOM0, SP0, KNUM25, KDEN26, KINT27, KFRC28, KSIM30, KCD31, ID, CINT, SBKO0, SBKC1}

R={

1. OPERATION -> **OPAR0** OPERAND\_ADD OPERATIONTALE
2. OPERATION -> **OPAR1** OPERAND\_ADD OPERATIONTALE
3. OPERATION -> OPERAND\_ADD OPERATIONTALE
4. OPERATIONTALE -> **OPAR0** OPERAND\_ADD OPERATIONTALE
5. OPERATIONTALE -> **OPAR1** OPERAND\_ADD OPERATIONTALE
6. OPERATIONTALE -> ε
7. OPERAND\_ADD -> OPERAND\_MUL OPERAND\_ADDTALE
8. OPERAND\_ADDTALE -> **OPMUL0** OPERAND\_MUL\_UNS OPERAND\_ADDTALE
9. OPERAND\_ADDTALE -> **OPMUL1** OPERAND\_MUL\_UNS OPERAND\_ADDTALE
10. OPERAND\_ADDTALE -> ε
11. OPERAND\_MUL\_UNS -> **OPAR0** OPERAND\_MUL
12. OPERAND\_MUL\_UNS -> **OPAR1** OPERAND\_MUL
13. OPERAND\_MUL\_UNS -> OPERAND\_MUL
14. OPERAND\_MUL -> SPECO **SPO0** OPERATION **SPC0**
15. OPERAND\_MUL -> **SPO0** OPERATION **SPC0**
16. OPERAND\_MUL -> OPCOMDEN **SPO0** OPERATION **SCOM0** OPERATION **SPC0**
17. OPERAND\_MUL -> **CINT**
18. OPERAND\_MUL -> **ID** OPERAND\_MUL\_TALE
19. OPERAND\_MUL\_TALE -> **SBKO0** ARRNUMS **SBKC1** OPERAND\_MUL\_VTALE
20. OPERAND\_MUL\_TALE -> OPERAND\_MUL\_VTALE
21. ARRNUMS -> ARRNUM ARRNUMTALE
22. ARRNUMTALE -> **SCOM0** ARRNUMS
23. ARRNUMTALE -> ε
24. ARRNUM -> OPERATION
25. OPERAND\_MUL\_VTALE -> **SP0** OPERAND\_MUL\_VTALESP0
26. OPERAND\_MUL\_VTALE -> ε
27. OPERAND\_MUL\_VTALESP0 -> **KNUM25**
28. OPERAND\_MUL\_VTALESP0 -> **KDEN26**
29. SPECO -> **KINT27**
30. SPECO -> **KFRC28**
31. SPECO -> **KSIM30**
32. OPCOMDEN -> **KCD31**

}

**3.2 Grammar logical expression**

GRAMMAR\_LOG={N,T,R,LE}

N={LE, DISO, DISOTALE, CONJO, CONJOTALE, NEGO, RATOP, RATIOTALE, RATIOTALETALE}

T={ KOR18, KAND19, KNOT, GRAMMAR\_OPERATION, OPCP1, OPCP2, OPCP4, OPCP5, OPCP6, OPINIT0}

R={

1. LE -> DISO DISOTALE
2. DISOTALE -> **KOR18** DISO
3. DISOTALE -> ε
4. DISO -> CONJO CONJOTAIL
5. CONJOTAIL -> **KAND19** CONJO
6. CONJOTAIL -> ε
7. CONJO -> **KNOT20** NEGO
8. CONJO -> NEGO
9. NEGO -> GRAMMAR\_OPERATION RATIOTALE
10. RATIOTALE -> RATOP GRAMMAR\_OPERATION RATIOTALE
11. RATIOTALE -> ε
12. RATIOTALETALE -> DISOTALE
13. RATIOTALETALE -> CONJOTALE
14. RATOP -> **OPCP2**
15. RATOP -> **OPCP4**
16. RATOP -> **OPCP5**
17. RATOP -> **OPCP6**
18. RATOP -> **OPINIT0**

}

* 1. **Grammar operator set**

GRAMMAR\_OPERATOR\_SET={N,T,R,OSET}

N={OSET, OSETTALE, O, OTALE, UNLO, UNOTCONDO, AO, FO, GO, INO, OUTO, COMO, UCONDO, MACHED, MACHEDTALE, MACHEDTALETALE, OPTIONAL, TALE, TALETALE , TALETALE TALE, NINO, INM, NOUTO, INMTALE, ARRNUMTALE, AOTALE, AOTALETALE, AOTALETALETALE }

T={ SSCOL0, SCOL0, KIF7, GRAMMAR\_LOG, KTH8, KEL9, OPA0, GRAMMAR\_OPERATION, KBG5, KEN6, SPO0, SPC0, KRD21, KRDL22, KWT23, KWLT24, KGO13, KFR10, KTO11, KDO12, ID, SBKO0, SBKC0, SCOM0, SP0, KNUM, KDEN }

R=

{

1. OSET -> O **SSCOL0** OSETTALE
2. OSETTALE -> OSET
3. OSETTALE -> ε
4. O -> **ID** OTALE
5. OTALE -> **SCOL0** UNLO
6. OTALE -> AOTALE
7. O-> UNLO
8. UNLO-> UCONDO
9. UNLO-> UNOTCONDO
10. UNOTCONDO -> FO
11. UNOTCONDO -> GO
12. UNOTCONDO -> INO
13. UNOTCONDO -> OUTO
14. UNOTCONDO -> COMO
15. UCONDO -> **KIF7** **GRAMMAR\_LOG** **KTH8** MACHED OPTIONAL
16. MACHED -> **KIF7** **GRAMMAR\_LOG** **KTH8** MACHED **KEL9** MACHED
17. MACHED -> UNOTCONDO
18. MACHED -> **ID** MATCHEDTALE
19. MATCHEDTALE -> **SCOL0** MATCHEDTALETALE
20. MATCHEDTALE -> AOTALE
21. MATCHEDTALETALE -> UNOTCONDO
22. MATCHEDTALETALE -> AO
23. OPTIONAL -> **KEL9** TALE
24. OPTIONAL -> ε
25. TALE -> **KIF7** **GRAMMAR\_LOG** **KTH8** TALE
26. TALE -> **ID** TALETALE
27. TALE -> UNOTCONDO
28. TALETALE -> **SCOL0** TALETALETALE
29. TALETALE -> AOTALE
30. TALETALETALE -> UNOTCONDO
31. TALETALETALE -> AO
32. AO -> **ID** AOTALE
33. AOTALE -> **OPA0** **GRAMMAR\_OPERATION**
34. AOTALE -> **SBKO0** **GRAMMAR\_OPERATION** ARRNUMTALE **SBKC1** **OPA0** **GRAMMAR\_OPERATION**
35. AOTALE -> SP0 AOTALETALE
36. AOTALETALE -> **KNUM** AOTALETALETALE
37. AOTALETALE -> **KDEN** AOTALETALETALE
38. AOTALETALETALE -> **OPA0** **GRAMMAR\_OPERATION**
39. AOTALETALETALE -> **SBKO0** **GRAMMAR\_OPERATION** ARRNUMTALE **SBKC1** **OPA0** **GRAMMAR\_OPERATION**
40. ARRNUMTALE -> **SCOM0** **GRAMMAR\_OPERATION** ARRNUMTALE
41. ARRNUMTALE -> ε
42. COMO -> **KBG5** OSET **KEN6**
43. INO -> NINO **SPO0** INM **SPC0**
44. NINO -> **KRD21**
45. NINO -> **KRDL22**
46. INM -> **ID** INMTALE
47. INMTALE -> **SBKO0** **GRAMMAR\_OPERATION** ARRNUMTALE **SBKC1**
48. INMTALE -> ε
49. OUTO -> NOUTO **SPO0** **GRAMMAR\_OPERATION** **SPC0**
50. NOUTO -> **KWT23**
51. NOUTO -> **KWLT24**
52. GO -> **KGO13** **ID**
53. FO -> **KFR10** **ID** **OPA0** **GRAMMAR\_OPERATION** **KTO11** **GRAMMAR\_OPERATION** **KDO12** O

}

**3.4 Grammar types**

GRAMMAR\_TYPES={N, E, R T}

N={T, ST, UT, ARR, IDIAPS, IDIAP, IDIAPSTALE, TDIAP, ENUM, IDSET, IDSETTALE, TDIAPTALE, TDIAPTALETALE, TDIAPTALETALETALE}

T={ID, KTINT0, KTRAT1, KARR3, SBKO0, SBKC0, KOF4, SCOM0, CINT, SPO0, SPC0, OPDPS0, OPAR0, OPAR1 }

R= {

1. T -> ST
2. T -> UT
3. UT -> **ID**
4. ST -> **KTINT0**
5. ST -> **KTRAT1**
6. ST -> ARR
7. ST -> TDIAP
8. ST -> ENUM
9. ARR -> **KARR3** **SBKO0** IDIAPS **SBKC0** **KOF4** T
10. IDIAPS -> IDIAP IDIAPSTALE
11. IDIAPSTALE -> **SCOM0** IDIAPS
12. IDIAPSTALE -> ε
13. IDIAP -> **CINT** **OPDPS0** **CINT**
14. TDIAP -> **OPAR0** TDIAPTALE
15. TDIAP -> **OPAR1** TDIAPTALE
16. TDIAP -> TDIAPTALE
17. TDIAPTALE -> **CINT** **OPDPS0** TDIAPTALETALE
18. TDIAPTALE -> **ID** **OPDPS0** TDIAPTALETALE
19. TDIAPTALETALE -> **OPAR0** TDIAPTALETALETALE
20. TDIAPTALETALE -> **OPAR1** TDIAPTALETALETALE
21. TDIAPTALETALE -> TDIAPTALETALETALE
22. TDIAPTALETALETALE -> **CINT**
23. TDIAPTALETALETALE -> **ID**
24. ENUM -> **SPO0** IDSET **SPC0**
25. IDSET -> **ID** IDSETTALE
26. IDSETTALE -> **SCOM0** IDSET
27. IDSETTALE -> ε

}

3.5. **Grammar variable description**

GRAMMAR\_VAR\_DISCRIPTION={N, T, R, DV}

N={DV, VSET, VSETTALE}

T={ SCOL0, GRAMMAR\_TYPES, SSCOL0, SCOM0, ID}

R={

1. DV -> VSET **SCOL0** **GRAMMAR\_TYPES** **SSCOL0**
2. VSET -> **ID** VSETTALE
3. VSETTALE -> **SCOM0** VSET
4. VSETTALE -> ε

}

**3.6 Grammar type description**

GRAMMA\_LAB\_DESCRIPTION={N, T, R, DL}

N={DL, LSET, LSETTALE}

T={ SSCOL0, ID , SCOM0}

R={

1. DL -> LSET **SSCOL0**
2. LSET -> **ID** LSETTALE
3. LSETTALE -> **SCOM0** LSET
4. LSETTALE -> ε

}

**3.7 Grammar const description**

GRAMMA\_CONST\_DESCRIPTION={N, T, R, DC}

N={ NC, IC, ICTALE}

T={ OPINIT0, SSCOL0, OPAR0, OPAR1, ID, CINT}

R={

1. DC -> **ID** **OPINIT0** IC **SSCOL0** DC
2. IC -> **OPAR0** ICTALE
3. IC -> **OPAR1** ICTALE
4. IC -> ICTALE
5. ICTALE -> **ID**
6. ICTALE -> **CINT**

}

**3.8 Grammar type description**

GRAMMA\_TYPE\_DESCRIPTION={N, T, R, DT}

N={DT }

T={ ID, OPINIT0, GRAMMAR\_TYPES, SSCOL0 }

R={

1. DT -> **ID** **OPINIT0** **GRAMMAR\_TYPES** **SSCOL0** DT
2. DT -> ε

}

**3.9 Main grammar**

GRAMMAR\_MAIN={ N, T, R, DT}

N={ N, SD, SO }

T={ KPR , SSCOL0, KBG , KEN , ID , GRAMMA\_CONST\_DESCRIPTION , GRAMMAR\_VAR\_DISCRIPTION , GRAMMA\_TYPE\_DESCRIPTION, GRAMMA\_LAB\_DESCRIPTION, KCON, KVAR, KTYP, KLAB, GRAMMAR\_OPERATOR\_SET, SP0 }

R={

1. S -> **KPR** N **SSCOL0** SD **KBG** SO **KEN** **SP0**
2. N -> **ID**
3. SD -> **KCON** **GRAMMA\_CONST\_DESCRIPTION** SD
4. SD -> **KVAR** **GRAMMAR\_VAR\_DISCRIPTION** SD
5. SD -> **KTYP** **GRAMMA\_TYPE\_DESCRIPTION** SD
6. SD -> **KLAB** **GRAMMA\_LAB\_DESCRIPTION** SD
7. SD -> ε
8. SO -> **GRAMMAR\_OPERATOR\_SET**

}

1. **Results and tests.**

**Program source:**

program hello;

var a: Integer;

begin

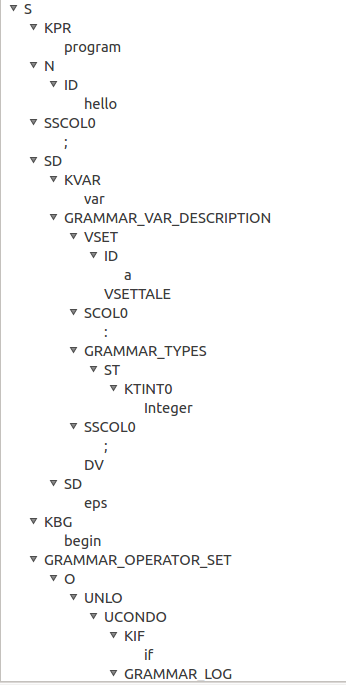
if a = -111 then a := a + a

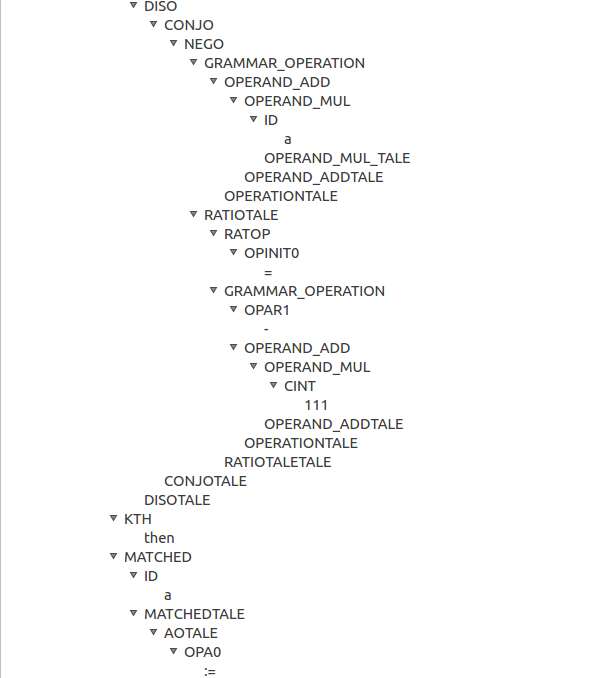
else b := b + b;

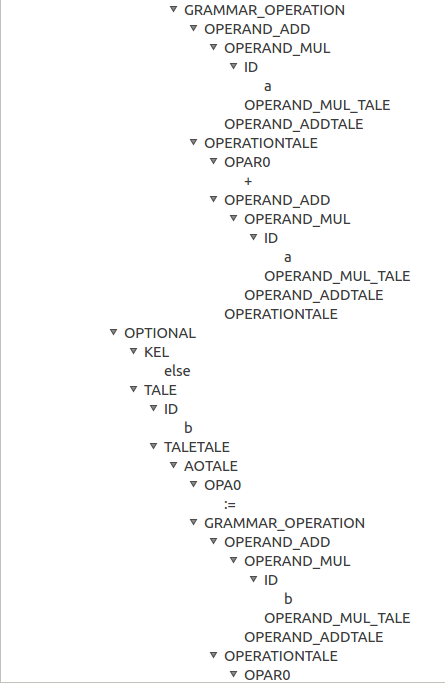
writeln(a.denominator);

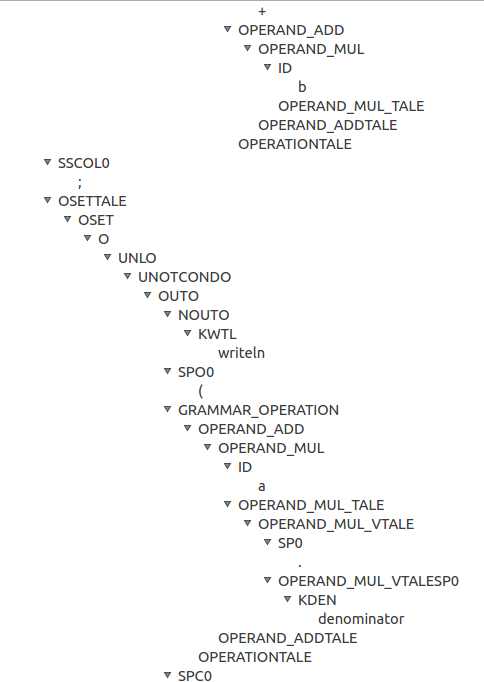
end.

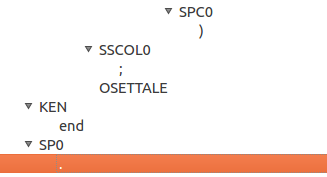
**Syntax tree.**

****

****

****

****

****