Прикладная программа

«Лексический / Синтаксический анализатор Delphi-подобного языка программирования»

Исходный код для изучения студентами направлений 09.03.01 и 09.03.04 по дисциплине «Теория автоматов и формальных языков»

Может применяться для изучения алгоритма лексического / синтаксического анализатора в среде C#

Разработал:

Галкин Роман Васильевич

Научный руководитель:

Наточая Елена Николаевна

ОГУ, 2019

Содержание

1 Формальное описание синтаксиса модельного языка программирования

1.1 Базовая грамматика языка

Синтаксис группы операций (в порядке следования: неравно, равно, меньше, меньше или равно, больше, больше или равно)
<операции_группы_отношения>:: = < > | = | < | <= | > | >=
Синтаксис группы операций (в порядке следования: сложение, вычитание, дизъюнкция)
<операции_группы_сложения>:: = + | - | or
Синтаксис группы операций (в порядке следования: умножение, деление, конъюнкция)
<операции_группы_умножения>::= * | / | and
Синтаксис операции
<унарная_операция>::= not
Структура программы
<программа>::= *program var* <описание> *begin* <оператор> {; <оператор>} *end*.
Синтаксис команд описания данных
<описание>::= <тип> <идентификатор> { , <идентификатор> }
Описание типов (в порядке следования: целый, действительный, логический)
<тип>::= *integer* | *real* | *boolean*
Синтаксис составного оператора
<составной>::= «[» <оператор> { (: | перевод строки) <оператор> } «]»
Оператор присваивания
<присваивания>::= <идентификатор> *as* <выражение>
Оператор условного перехода
<условный>::= *if* <выражение> *then* <оператор> [*else* <оператор>]
Оператор цикла с фиксированным числом повторений
<фиксированного_цикла>::= *for* <присваивания> *to* <выражение> *do* <оператор>
Синтаксис условного оператора цикла
<условного_цикла>::= *while* <выражение> *do* <оператор>
Синтаксис оператора ввода
<ввода>::= *read* «(»<идентификатор> {, <идентификатор> } «)»
Синтаксис оператора вывода

<вывода>::= *write* «(»<выражение> {, <выражение> } «)»	
Признак начала комментария	Признак конца комментария
{	}

1.2 Формальная грамматика (РБНФ)

Опишем с помощью формальных грамматик синтаксис модельного языка *MOD*. Грамматика будет иметь правила вывода вида:

PRO → program D2; BOD.
D2 → var D1
D1 → DIM | D1; DIM
DIM → integer I1 | bool I1 | real I1
I1 → IND | I1, IND
BOD → begin OPR end
S1 → OPR | S1: OPR | S1¬ OPR
E2 → EXP | E2, EXP
OPR → [S1] | if EXP then OPR else OPR | if EXP then OPR | while EXP do OPR | for IND as EXP to EXP do OPR | read(I1) | write(E2) | IND as EXP | OPR
EXP → E1 | E1=E1 | E1>E1 | E1<E1 | E1<>E1 | E1>=E1 | E1<=E1
El → MUL | MUL+E1 | MUL-E1 | MUL or El
MUL → FACT | FACT*MUL | FACT/MUL | FACT and MUL
FACT → IND | NUM | LOG | not FACT | (EXP)
LOG → true | false
IND → CHR | IND CHR | IND RND
CHR → 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
DIG → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
NUM → INT | REAL
INT → BIN | OCT | DEC | HEX
BINT → 0 | 1
BIN → BINT B | BINT b | BINT BIN
OCTI → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
OCT → OCTI O | OCTI o | OCTI OCT
DEC → DIG D | DIG d | DIG | DEC
HCH → a | b | c | d | e | f | A | B | C | D | E | F
HEX → DIG HCH | HEX
LDIG → DIG | LDIG DIG
REAL → LDIG POR | LDIG.LDIG POR | .LDIG POR | .LDIG | LDIG.LDIG
POR → E+LDIG | E-LDIG | e+LDIG | E-LDIG | E LDIG | e LDIG

1.3 Синтаксис языка

1.3.1 Формы Бэкуса-Наура модельного языка

Синтаксис модельного языка *M* с помощью форм Бэкуса-Наура будет выглядеть следующим образом:

<ключевое_слово> ::= program | var | begin | end | integer | real | bool | read | write | if
| then | else | while | do | true | false | to | or | and | not | as | for
<разделитель> ::= (|) | , | . | ; | : | ¬ | < > | = | < | <= | > | >= | + | - | * | / | { | } | [|]
<число> ::= {/< цифра> /}
<программа> ::= *program* <описание> ; <тело>.
<описание>::= var (*integer* | *real* | *bool*) <идентификатор> { , <идентификатор> }
<тело> ::= *begin* {<оператор>; } *end*
<оператор> ::= <присваивания> | <условный> | <условного_цикла> | <фиксированного_цикла> | <составной> | <ввода> | <вывода>
<присваивания> ::= <идентификатор> as <выражение>
<условный> ::= *if* <выражение> *then* <оператор> *else* <оператор>
<условный_цикла> ::= *while* <выражение> *do* <оператор>
<фиксированного_цикла> ::= for <присваивания> to <выражение> do <оператор>
<составной>::= «[» <оператор> { (: | перевод строки) <оператор> } «]»
<ввода>:: = *read*(<идентификатор>)
<вывода>:: = *write*(<выражение>)

Выражения языка задаются правилами:

<выражение>:: = <сумма> | <сумма> (= | < | > | >= | < > | <=) <сумма>
<сумма> ::= <произведение> { (+ | - | and) <произведение>}
<произведение>:: = <множитель> { (* | / | or) <множитель>}
<множитель>:: = <идентификатор> | <число> | <логическая_константа> | not <множитель> | (<выражение>)
<логическая_константа>:: = *true* | *false*

Правила, определяющие идентификатор, букву и цифру:

<идентификатор> ::= <буква> { <буква> | <цифра> }
<буква> ::= *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*
<цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

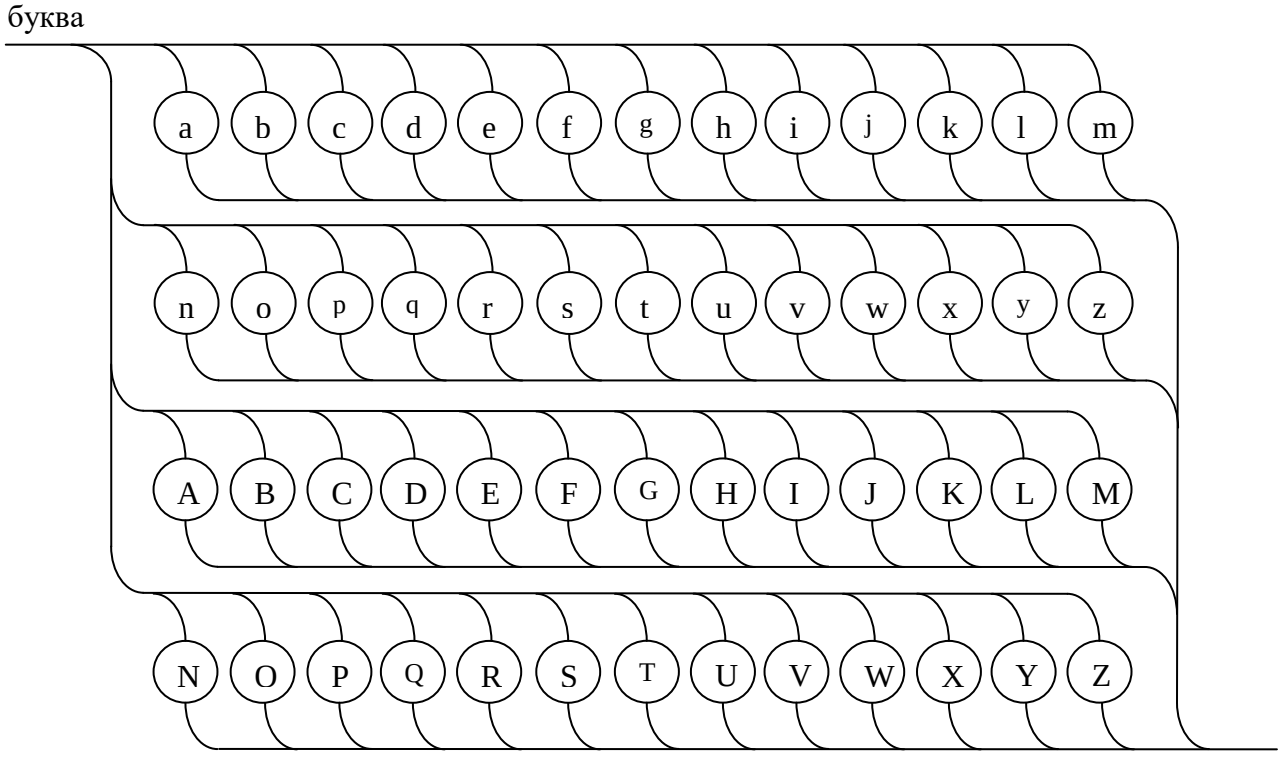
Правила, определяющие целые числа:
<целое>::= <двоичное> | <восьмеричное> | <десятичное> | <шестнадцатеричное>
<двоичное>::= {/ 0 | 1 /} (*B* | *b*)

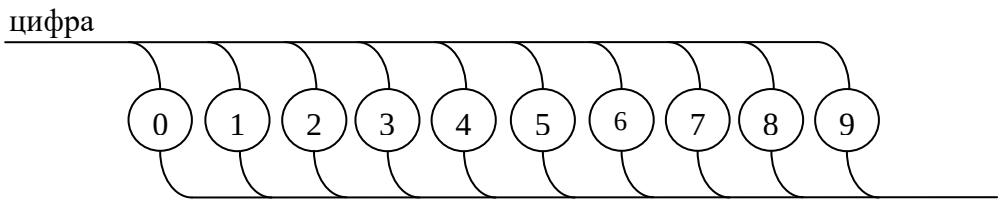
<восьмеричное>::= {/ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 /} (*O* | *o*)
<десятичное>::= {/ <цифра> /} [*D* | *d*]
<шестнадцатеричное>::= <цифра> {<цифра> | *A* | *B* | *C* | *D* | *E* | *F* | *a* | *b* | *c* | *d* | *e* | *f*} (*H* | *h*)

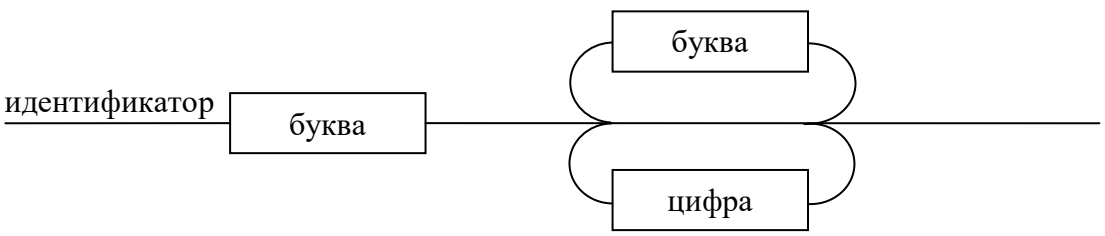
Правила, описывающие действительные числа:
<действительное>::= <числовая_строка> <порядок> | [<числовая_строка>] . <числовая_строка> [порядок]
<числовая_строка>::= {/ <цифра> /}
<порядок>::= (E | e)[+ | -] <числовая_строка>

1.3.2 Диаграммы Вирта

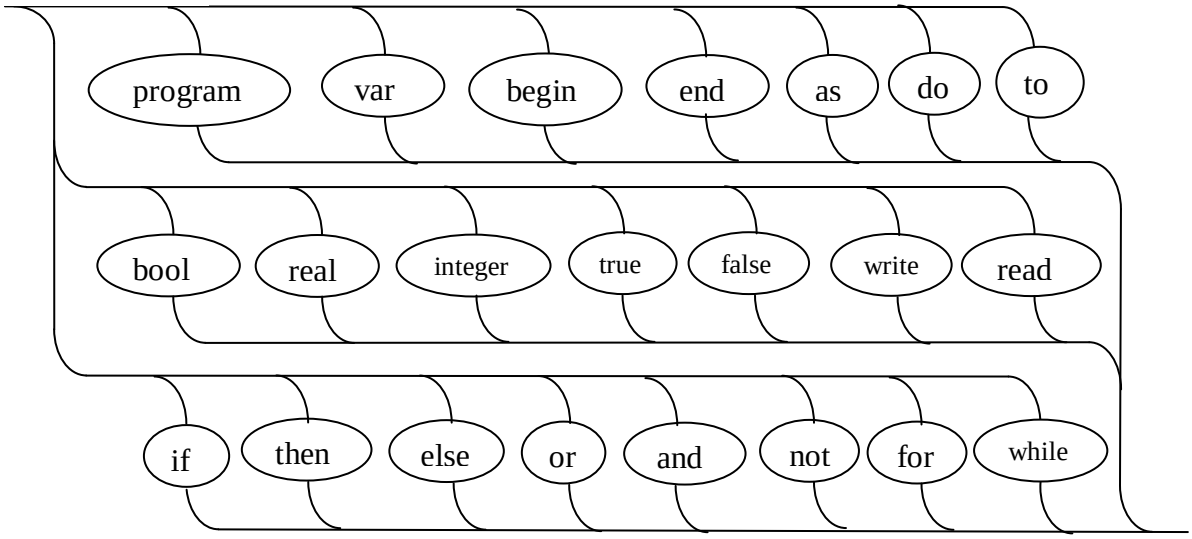
Описание синтаксиса модельного языка *M* с помощью диаграмм Вирта представлено на следующих рисунках:



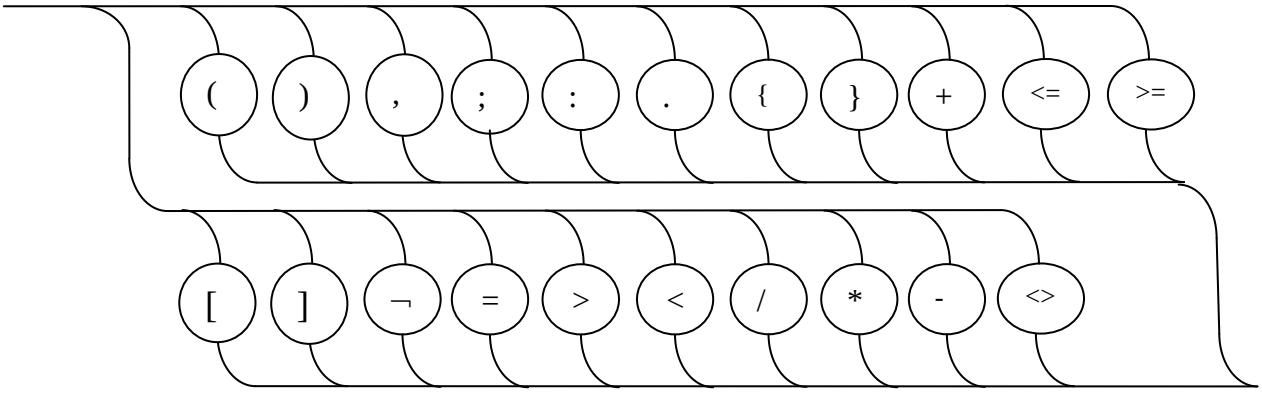




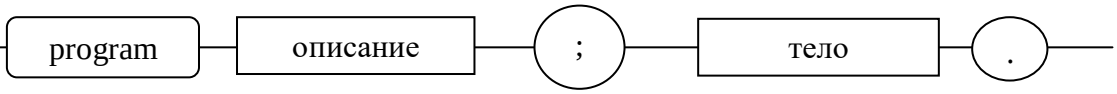
ключевое_слово



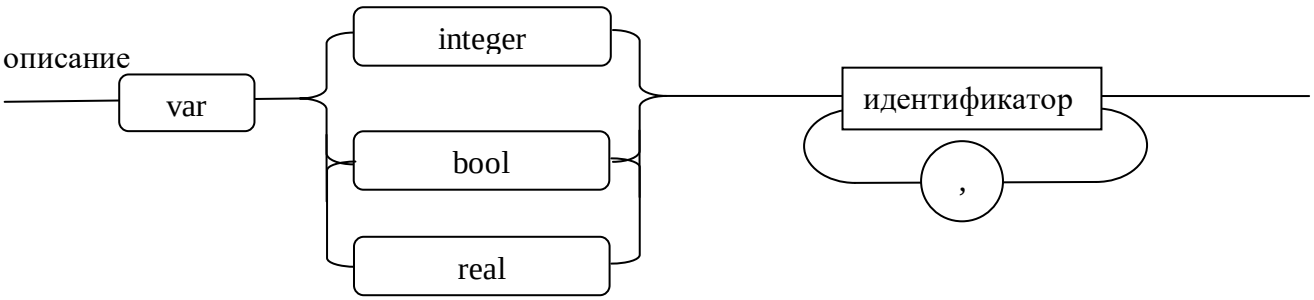
разделитель



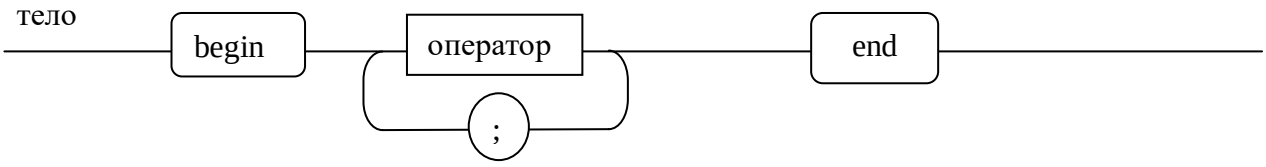
программа

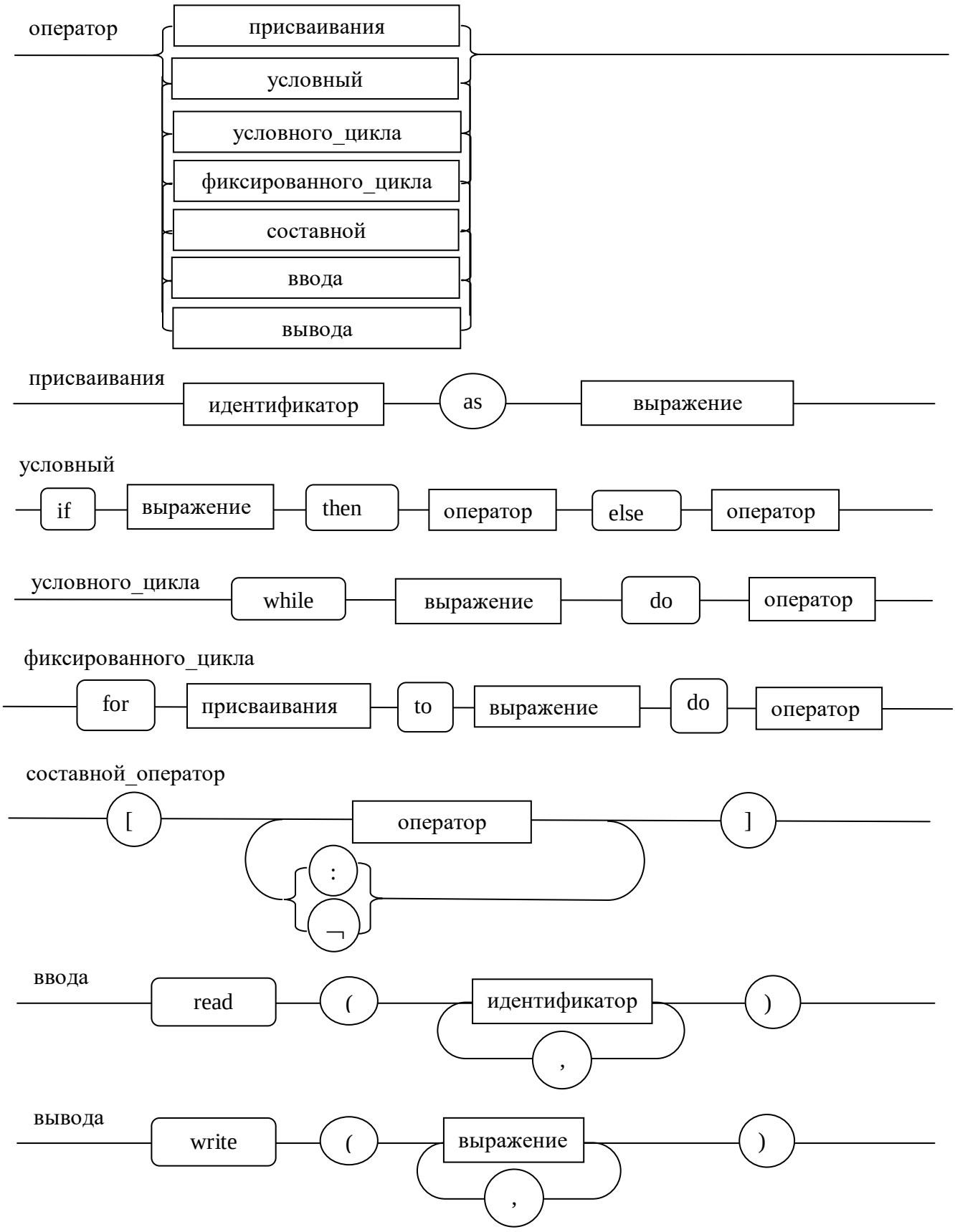


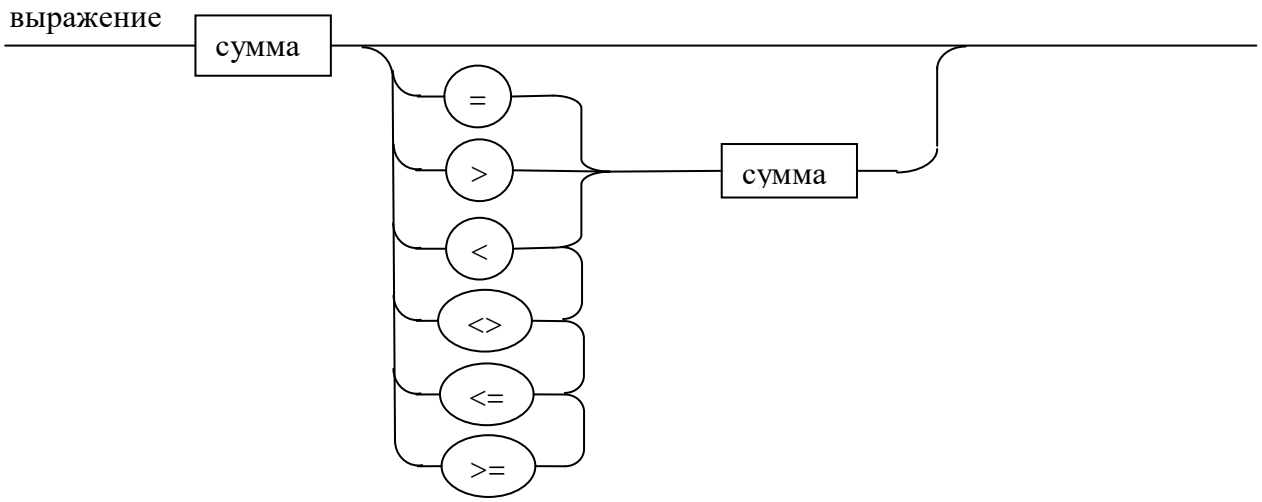
описание

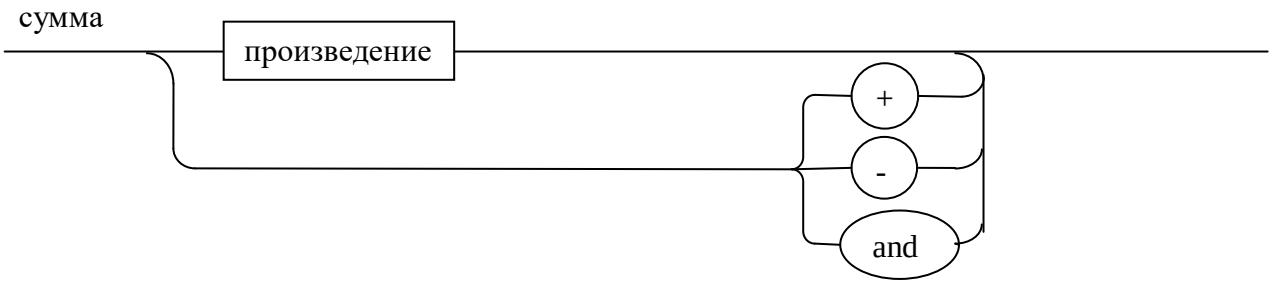


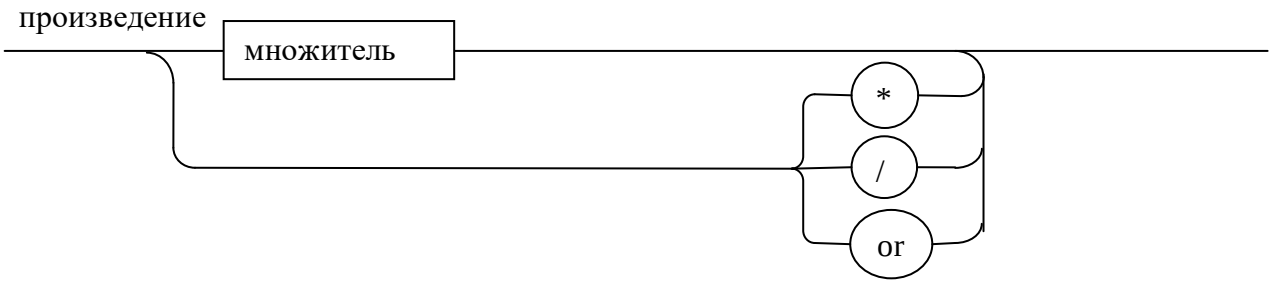
тело

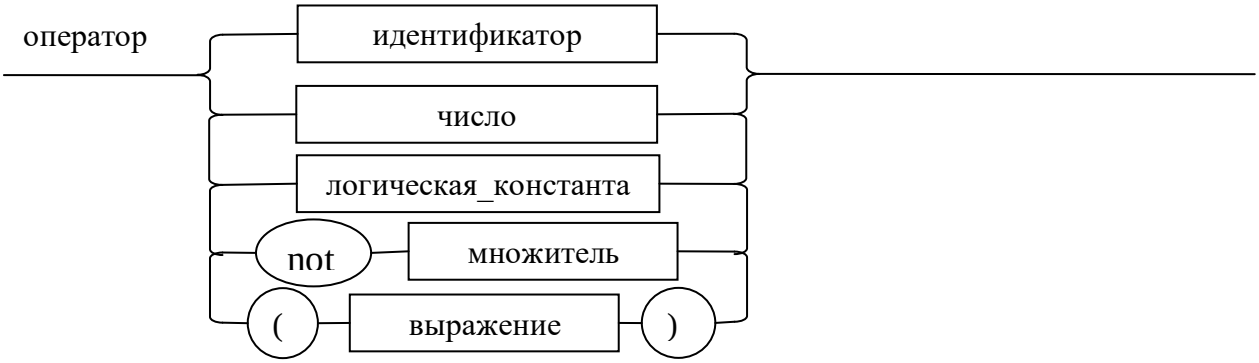



оператор
присваивания
условный
условного_цикла
фиксированного_цикла
составной
ввода
вывода
присваивания
идентификатор
as
выражение
условный
if
выражение
then
оператор
else
оператор
условного_цикла
while
выражение
do
оператор
фиксированного_цикла
for
присваивания
to
выражение
do
оператор
составной_оператор
[
оператор
]
:
¬
ввода
read
(
идентификатор
)
,
вывода
write
(
выражение
)
,


выражение
сумма
=
>
<
<>
<=
>=
сумма


сумма
произведение
+
-
and

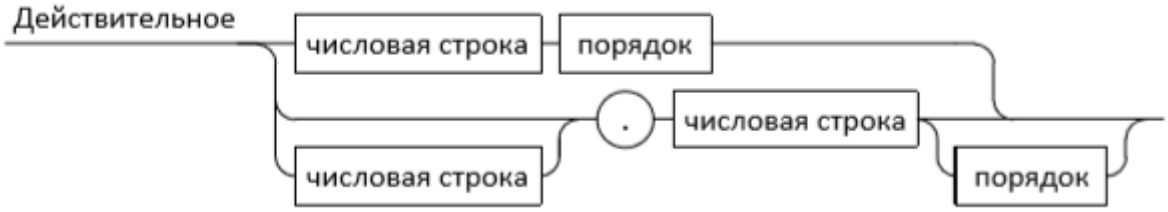

произведение
множитель
*
/
or

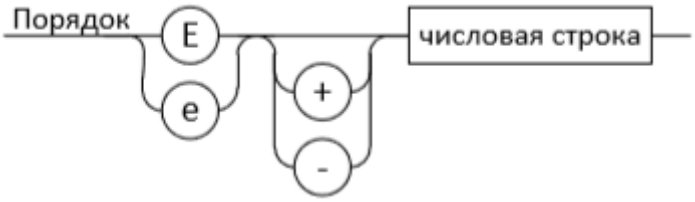

оператор
идентификатор
число
логическая_константа
not
множитель
(
выражение
)


логическая константа
true
false

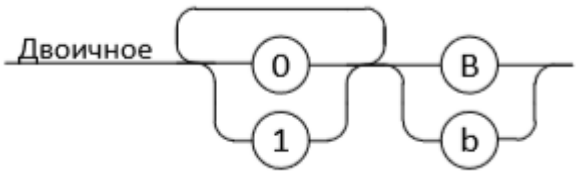

Число
действительное
целое
Числовая строка
цифра

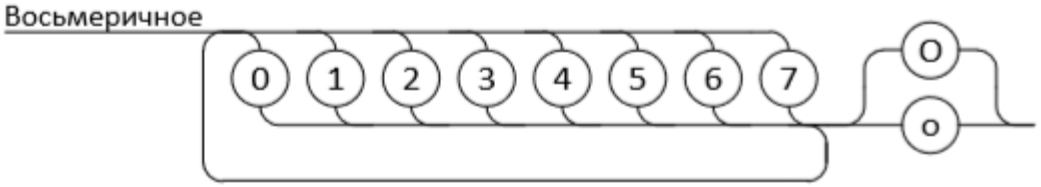

Целое
двоичное
восьмеричное
десятичное
шестнадцатиричное

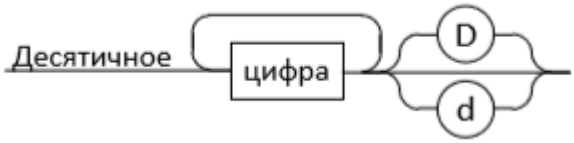

Действительное
числовая строка
порядок
.
числовая строка
числовая строка
порядок

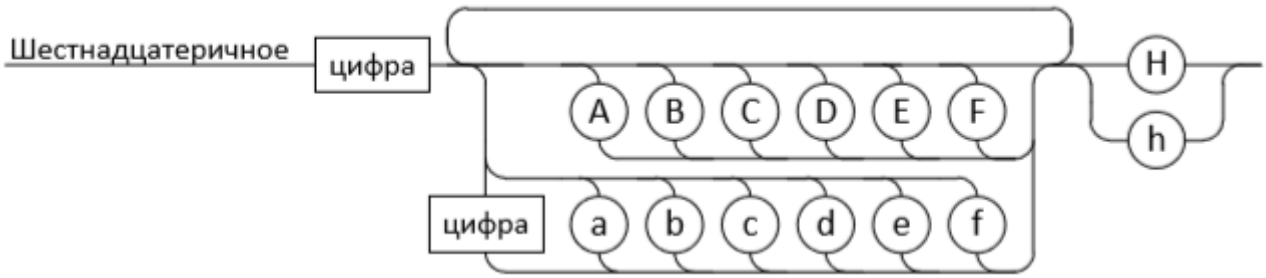

Порядок
E
e
+
-
числовая строка


Числовая строка
цифра

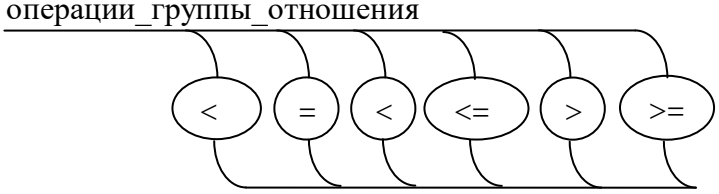

Двоичное
0
1
B
b


Восьмеричное
0
1
2
3
4
5
6
7
O
o

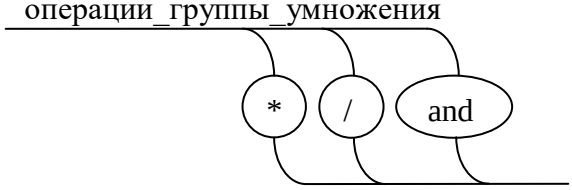

Десятичное
цифра
D
d


Шестнадцатеричное
цифра
A
B
C
D
E
F
H
h
цифра
a
b
c
d
e
f

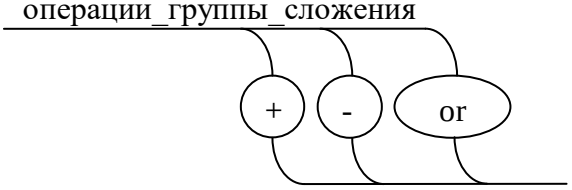
операции_группы_отношения



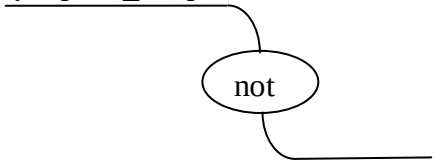
операции_группы_умножения



операции_группы_сложения



унарная_операция



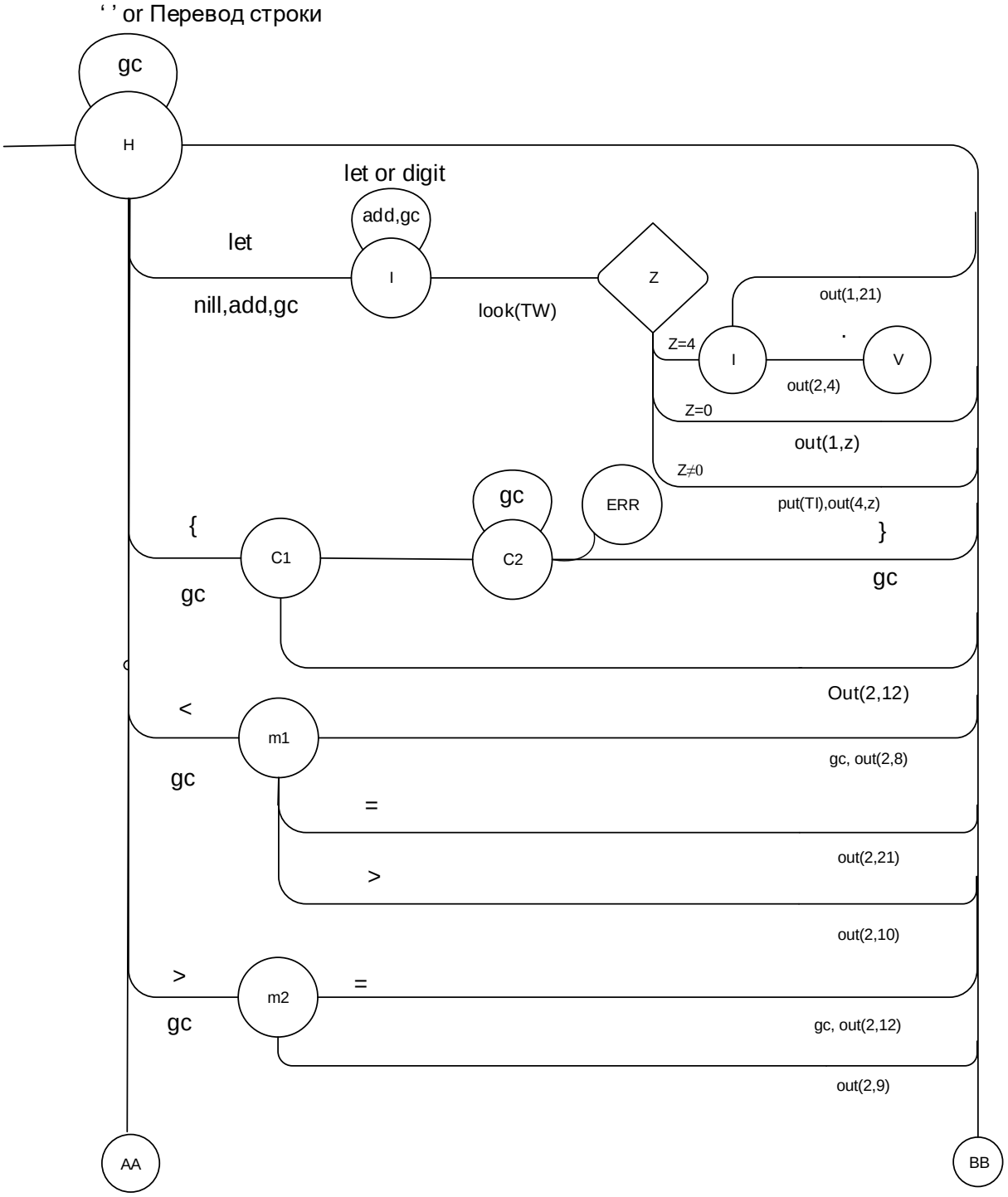
2 Конечный автомат

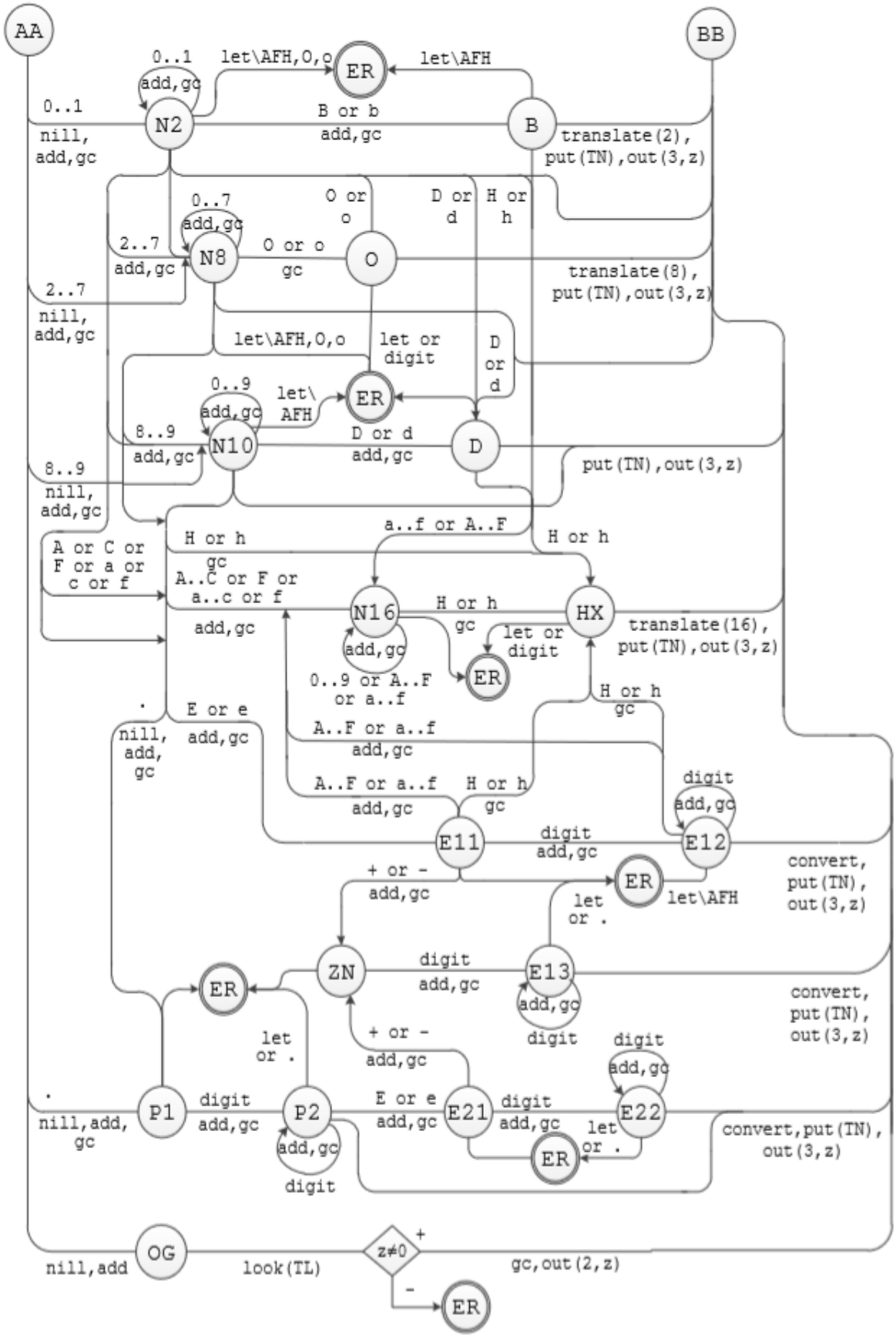
2.1 Таблица лексем

Таблица	Лексемы
Разделители	(|) | , | . | ; | : | \n | < | > | = | <= | >= | + | - | * | / | { | } | [|] | <>
Служ. слова	program | var | begin | end | integer | real | bool | read | write | if | then | else | while | do | true | false | to | or | and | not | as | for

Таблицы идентификаторов и чисел формируются в ходе лексического анализа.

2.2 Диаграммы состояний (с действиями)





2.3 Набор контрольных примеров

2.3.1 Программа на модельном языке M, вычисляющая среднее арифметическое чисел, введенных с клавиатуры

```
1  | program var
2  | integer k, n, sum, i;
3  | begin
4  |         read(n);
5  |         sum as 0;
6  |         i as 1;
7  |         while i<=n do [read(k): sum as sum + k];
8  |         {Используем составной оператор в теле цикла, разделяя операторы «:»}
9  |         write(sum/n);
10 | end.
```

2.3.2 Программа считающая минимум из чисел, введённых с клавиатуры

```
1  | program var
2  | integer min,n,k,i;
3  | begin
4  |         read(n);
5  |         read(k);
6  |         min as k;
7  |         {считали первое значение, чтобы сравнивать с ним}
8  |         for i as 2 to n do [read(k) {Используем составной оператор}
9  |                 if k < min then {Используем оператор присваивания}
10 |                         min as k
11 |         ]; {закрываем составной оператор}
12 |         write(min);
13 | end.
```

2.3.3 Программа вычисления площади круга

```
1  | program var
2  | real Pi, r, s;
3  | begin
4  |         read(r);
5  |         S as Pi * r * r;
6  |         write(S);
7  | end.
```

2.3.4 Программа упорядочения двух значений по возрастанию

```
1  | program var
2  | integer x,y,v;
3  | begin
4  |         read(x);
5  |         read(y);
6  |         if x>y then[v as x: x as y: y as v];
7  |         write(x);
```

```
8       write(y);
9   end.
```

2.3.5 Программа, вычисляющая необходимое количество чисел Фибоначчи

```
1   program var
2   integer a,b,c,i,n;
3   begin
4       a as 1;
5       b as 1;
6       read(n); {считываем необходимое количество чисел}
7       for i as 3 to n do[c as a + b
8           write(c)
9           a as b
10          b as c
11          {Нет использования «;», так как в составном операторе,
12          после оператора идёт «¬» или «:»}
13      ];
14  end.
```

2.3.6 Программа, выдающая значение текущего сезона

(0 – Зима, 1 – Весна, 2 – Лето, 3 – Осень).

```
1   program var
2   integer m,s;
3   begin
4       read(m);
5       if (m=1) or (m=2) or (m=12) then
6           s as 0 {оператор присваивания}
7       else {составной оператор}
8       [if (m=3) or (m=4) or (m=5) then s as 1 {оператор присваивания}
9           else {составной оператор}
10          [if (m=6) or (m=7) or (m=8) then s as 2 {оператор присваивания}
11              else s as 3 {оператор присваивания}
12          ]
13      ];
14      {система вложенных друг в друга составных операторов}
15      {используются разделители «:» и «¬», непосредственно в самих
16  операторах}
        write(s);
    end.
```

3 Синтаксический анализатор

3.1 Разбор свойств и формализации грамматики модельного языка

Для следующего модельного языка:

```
1 | program var
2 | integer i;
3 | begin
4 |         read(i);
5 |         write(i*2);
6 | end.
```

построим цепочку левостороннего и правостороннего вывода простейшей программы на модельном языке из начального символа грамматики.

3.1.1 Цепочка левостороннего вывода

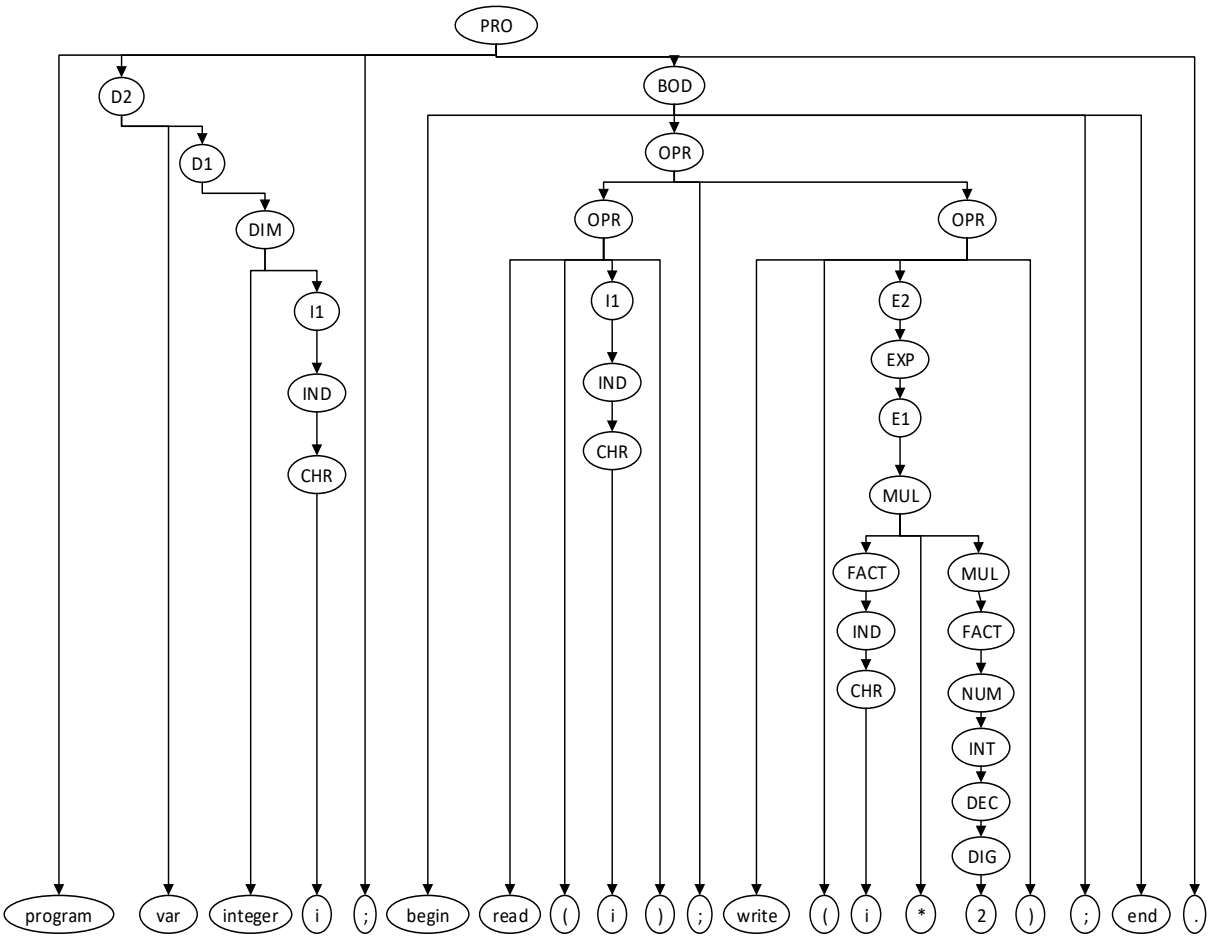
PRO → program D2; BOD. → program var D1; BOD. → program var DIM; BOD. → program var integer I1; BOD. → program var integer IND; BOD. → program var integer CHR; BOD. → program var integer i; BOD. → program var integer i; begin OPR; end. → program var integer i; begin OPR; OPR; end. → program var integer i; begin read(I1); OPR; end. → program var integer i; begin read(IND); OPR; end. → program var integer i; begin read(CHR); OPR; end. → program var integer i; begin read(i); OPR; end. → program var integer i; begin read(i); write(E2); end. → program var integer i; begin read(i); write(EXP); end. → program var integer i; begin read(i);write(E1); end. → program var integer i;begin read(i);write(MUL); end. → program var integer i;begin read(i); write(FACT*MUL); end. → program var integer i; begin read(i); write(IND*MUL); end. → program var integer i; begin read(i); write(CHR*MUL); end. → program var integer i ;begin read(i); write(i*MUL); end. → program var integer i; begin read(i); write(i*FACT); end. → program var integer i; begin read(i); write(i*NUM); end. → program var integer i; begin read(i); write(i*INT); end. → program var integer i; begin read(i); write(i*DEC); end. → program var integer i; begin read(i); write(i*DIG); end. → program var integer i; begin read(i); write(i*2); end.

3.1.2 Цепочка правостороннего вывода

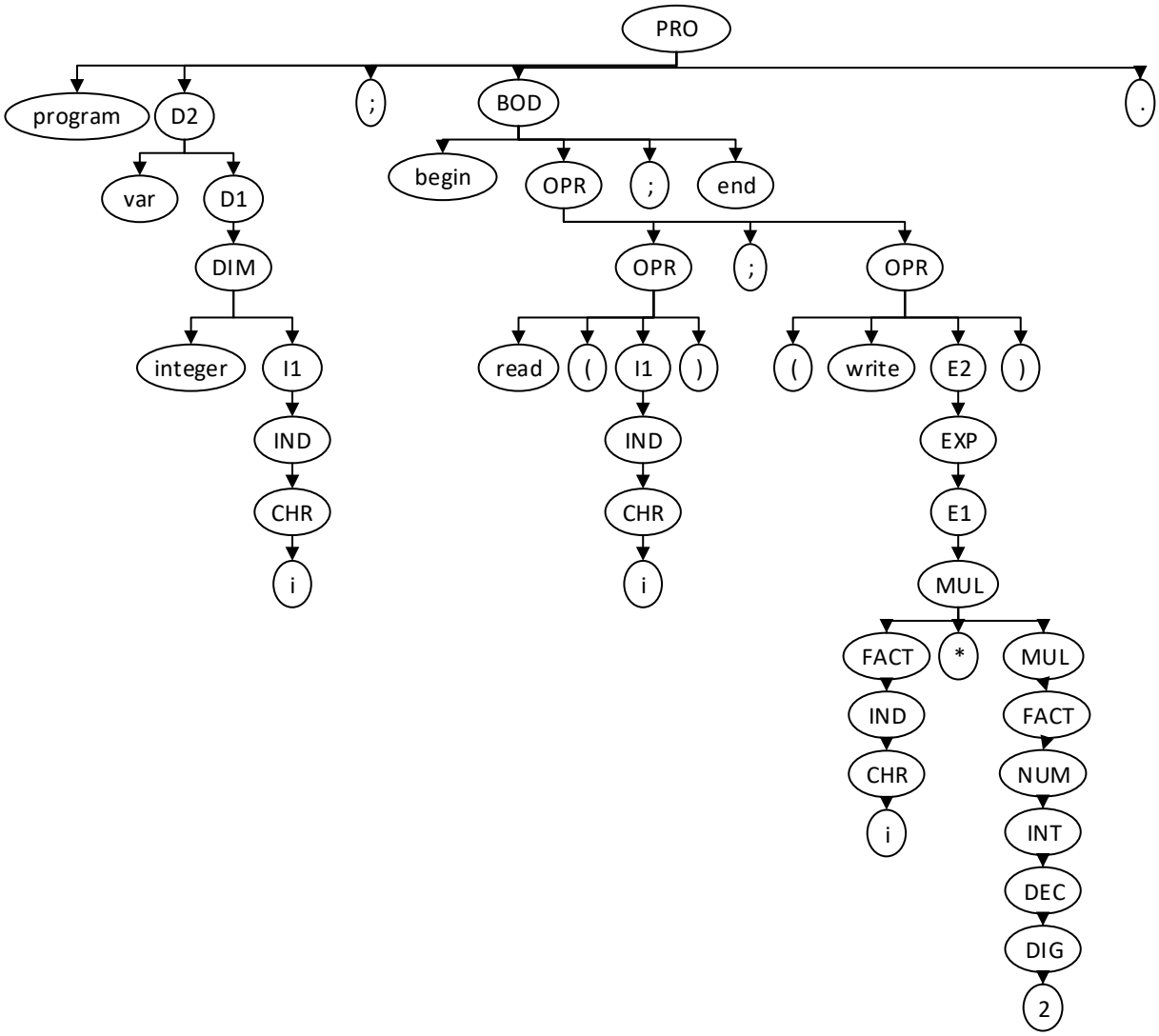
PRO → program D2; BOD. → program D2; begin OPR; end. → program D2; begin OPR; OPR; end. → program D2; begin OPR; write(E2); end. → program D2; begin OPR; write(EXP); end. → program D2; begin OPR; write(E1); end. → program D2; begin OPR; write(MUL); end. → program D2; begin OPR; write(FACT*MUL); end. → program D2; begin OPR; write(FACT*FACT); end. → program D2; begin OPR; write(FACT*NUM); end. → program D2; begin OPR; write(FACT*INT); end. → program D2; begin OPR; write(FACT*DEC); end. → program D2; begin OPR; write(FACT*DIG); end. → program D2; begin OPR; write(FACT*2); end. → program

D2; begin OPR; write(IND*2); end. → program D2; begin OPR; write(CHR*2); end. → program D2; begin OPR; write(i*2); end. → program D2; begin read(I1); write(i*2); end. → program D2; begin read(IND); write(i*2); end. → program D2; begin read(CHR); write(i*2); end. → program D2; begin read(i); write(i*2); end. → program var D1; begin read(i); write(i*2); end. → program var DIM; begin read(i); write(i*2); end. → program var integer I1; begin read(i); write(i*2); end. → program var integer IND; begin read(i); write(i*2); end. → program var integer CHR; begin read(i); write(i*2); end. → program var integer i; begin read(i); write(i*2); end.

3.2 Восходящее дерево разбора



3.3 Нисходящее дерево разбора


PRO
program
D2
;
BOD
.
var
D1
begin
OPR
;
end
DIM
OPR
;
OPR
integer
I1
read
(
I1
)
(
write
E2
)
IND
IND
EXP
CHR
CHR
E1
i
i
MUL
FACT
*
MUL
IND
FACT
CHR
NUM
i
INT
DEC
DIG
2

Список используемой литературы:

1 Введение в теорию алгоритмических языков и компиляторов: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева. - М.: ИД ФОРУМ, 2011. - 176 с. - ISBN 978-5-8199-0404-6. - Режим доступа: http://znanium.com/catalog.php?bookinfo=265617.

2 Ишакова Е.Н. Теория языков программирования и методов трансляции: учебное пособие. – Оренбург: ИПК ГОУ ОГУ, 2007. – 137 с.

3 Соколов А.П. Системы программирования: теория, методы, алгоритмы: Учеб. пособие. – М.: Финансы и статистика, 2004. – 320с.

4 Серебряков В.А. и др. Теория и реализация языков программирования: учебное пособие. – М.: МЗ-Пресс, 2003. – 345 с.