# **ОПИСАНИЕ ЛЕКСЕМ**

1. Идентификатор (имя);

2. Целое число;

3. Вещественное число;

4. «==» – операция сравнения (равно);

5. «!=» – операция сравнения (не равно);

6. «>=» – операция сравнения (больше или равно);

7. «<=» – операция сравнения (меньше или равно);

8. «||» – двойная вертикальная черта (логическое ИЛИ);

9. «&&» – двойной амперсанд (логическое И);

10. «if» – служебное слово (начало ветвления);

11. «else» – служебное слово (альтернативный вариант в ветвлении);

12. «while» – служебное слово (начало цикла);

13. «scan» – служебное слово (чтение данных из потока ввода/консоли);

14. «print» – служебное слово (запись данных в поток вывода/консоль);

15. «abs» – служебное слово (функция абсолютного значения/модуля числа);

16. «sqrt» – служебное слово (функция квадратного корня из числа);

17. «ln» – служебное слово (функция взятия натурального логарифма от числа);

18. «lg» – служебное слово (функция взятия логарифма по основанию 10 от числа);

19. «exp» – служебное слово (функция экспоненты от числа);

20. «int» – служебное слово (объявление целочисленной переменной);

21. «real» – служебное слово (объявление вещественной переменной);

22. «Ʇ» – символ терминала (окончание программы);

33. «!» – восклицательный знак (логическое НЕ);

40. «(» – открывающая круглая скобка (сложные выражения, условные выражения, аргументы функций);

41. «)» – закрывающая круглая скобка;

42. «\*» – умножение (операция произведения);

43. «+» – плюс (операция сложения);

44. «,» – запятая (разделение значений при инициализации массива или переменных при описании);

45. «-» – минус (операция вычитания, унарный минус);

47. «/» – деление (операция частного);

59. «;» – точка с запятой (окончание выражения);

60. «<» – операция сравнения (меньше);

61. «=» – равно (операция присваивания);

62. «>» – операция сравнения (больше);

91. «[» – открывающая квадратная скобка (индексация и инициализация массива);

93. «]» – закрывающая квадратная скобка;

123. «{» – открывающая фигурная скобка (блок выражений, инициализация массива);

125. «}» – закрывающая фигурная скобка;

# **КС-ГРАММАТИКА**

A – начальный нетерминал; a, k – имя переменной и константа.

A → int P;A | real P;A | aH=SZ;A | if(L){A}EZA | while(L){A}A | scan(aH);A | print(S);A | λ

P → aBQ

Q → ,aBQ | λ

B → =SZ | [k]C | λ

C → ={SD} | λ

D → ,SD | λ

S → S+T | S-T | T

T → T\*F | T/F | F

F → (S) | -GZ | abs(S) | sqrt(S) | ln(S) | lg(S) | exp(S) | aH | k

G → (S) | abs(S) | sqrt(S) | ln(S) | lg(S) | exp(S) | aH | k

H → [S] | λ

L → L||M | M

M → M&&N | N

N → (L) | !(L) | K

K → S<SZ | S<=SZ | S!=SZ | S==SZ | S>=SZ | S>SZ

E → else{A} | λ

Z → λ

# **КС-ГРАММАТИКА БЕЗ ЛЕВОЙ РЕКУРСИИ**

A → int P;A | real P;A | aH=SZ;A | if(L){A}EZA | while(L){A}A | scan(aH);A | print(S);A | λ

P → aBQ

Q → ,aBQ | λ

B → =SZ | [k]C | λ

C → ={SD} | λ

D → ,SD | λ

S → TU

U → +TU | -TU | λ

T → FV

V → \*FV | /FV | λ

F → (S) | -GZ | abs(S) | sqrt(S) | ln(S) | lg(S) | exp(S) | aH | k

G → (S) | abs(S) | sqrt(S) | ln(S) | lg(S) | exp(S) | aH | k

H → [S] | λ

L → MW

W → ||MW | λ

M → NX

X → &&NX | λ

N → (L) | !(L) | S<SZ | S<=SZ | S!=SZ | S==SZ | S>=SZ | S>SZ

E → else{A} | λ

Z → λ

# **КС-ГРАММАТИКА В НЕСТРОГОЙ ФОРМЕ ГРЕЙБАХ**

A → int P;A | real P;A | aH=SZ;A | if(L){A}EZA | while(L){A}A | scan(aH);A | print(S);A | λ

P → aBQ

Q → ,aBQ | λ

B → =SZ | [k]C | λ

C → ={SD} | λ

D → ,SD | λ

­

S → (S)VU | -GVU | abs(S)VU | sqrt(S)VU | ln(S)VU | lg(S)VU | exp(S)VU | aHVU | kVU

U → +TU | -TU | λ

T → (S)V | -GV | abs(S)V | sqrt(S)V | ln(S)V | lg(S)V | exp(S)V | aHV | kV

V → \*FV | /FV | λ

F → (S) | -GZ | abs(S) | sqrt(S) | ln(S) | lg(S) | exp(S) | aH | k

G → (S) | abs(S) | sqrt(S) | ln(S) | lg(S) | exp(S) | aH | k

H → [S] | λ

L → (L)XW | !(L)XW | -GVUOXW | abs(S)VUOXW | sqrt(S)VUOXW | ln(S)VUOXW | lg(S)VUOXW | exp(S)VUOXW | aHVUOXW | kVUOXW

W → ||MW | λ

M → (L)X | !(L)X | -GVUOX | abs(S)VUOX | sqrt(S)VUOX | ln(S)VUOX | lg(S)VUOX | exp(S)VUOX | aHVUOX | kVUOX

X → &&NX | λ

N → (L) | !(L) | -GVUO | abs(S)VUO | sqrt(S)VUO | ln(S)VUO | lg(S)VUO | exp(S)VUO | aHVUO | kVUO

O → <SZ | <=SZ | !=SZ | ==SZ | >=SZ | >SZ

E → else{A} | λ

Z → λ

# **СПИСОК ОПЕРАЦИЙ ОПС**

«=» – присваивание;

«+» – сложение;

«-» – вычитание;

«\*» – умножение;

«/» – деление;

«||» – логическое ИЛИ;

«&&» – логическое И;

«!» – логическое НЕ;

«<» – строгое меньше;

«<=» – нестрогое меньше;

«!=» – не равно;

«==» – равно;

«>=» – нестрогое больше;

«>» – строгое больше;

-‘ – унарный минус;

**s** – запись значения с потока ввода в переменную;

**p** – вывод в поток вывода;

**m** – модуль (абсолютное значение числа);

**r** – квадратный корень из числа;

**ln** – натуральный логарифм;

**lg** – десятичный логарифм;

**e** – экспонента;

**i** – индексация массива;

**jf** – переход при условии false;

**j** – безусловный переход;

**ar** – инициализация массива;

a – запись терминала в ОПС;

k – запись константы в ОПС;

Семантические программы. k – номер генерируемого элемента ОПС.

Программа **1**:

1. В магазин меток записывается k.

2. В ОПС записывается пустой элемент – место для будущей метки.

3. В ОПС записывается операция **jf** – переход при условии false.

Программа **2**:

1. Через верхний элемент магазина меток записывается k+2.

2. В магазин меток записывается k.

3. В ОПС записывается пустой элемент – место для будущей метки.

4. В ОПС записывается операция **j** – безусловный переход.

Программа **3**:

1. Через верхний элемент магазина меток, записывается k.

Программа **4**:

1. В магазин меток записывается k.

Программа **5**:

1. Через верхний элемент магазина меток записывается k + 2.

2. В ОПС записывается метка, значение которой читается из магазина меток.

3. В ОПС записывается операция **j** – безусловный переход.

Программа **6**:

1. Проверяем наличие переменной с таким именем в соответствующей таблице. Если имя есть, выводим ошибку.

2. Если нет, то добавляем в соответствующую таблицу переменных.

3. В магазин меток записывается k.

4. В ОПС записывается пустой элемент – место будущей переменной (ну или как-то здесь нужно зафиксировать её имя).

Программа **7**:

1. Через верхний элемент массива меток выделяем память для переменной и записываем в ОПС её адрес.

Программа **8**:

1. Через верхний элемент массива меток выделяем память для массива размерности, равной константе, расположенной в ОПС на следующей позиции после пустого элемента (с проверкой на целочисленность размерности) и в ОПС на заранее заготовленное место записываем адрес паспорта данного массива.

2. В ОПС записываем метку на данный адрес массива.

3. В ОПС записываем операцию **ar** – инициализация массива.

Программа **9**:

1. Через верхний элемент массива меток проверяем находится ли там пустой элемент. Если он не пустой, то ничего не делаем.

2. Если пустой, то проверяем разницу между текущим k и значением верхнего элемента массива меток. Если эта разница равна 0, то выделяем память для переменной, после чего удаляем пустой элемент из ОПС.

3. Если разница равна 1, то выделяем память для массива размерности, равной константе, расположенной в ОПС на следующей позиции после пустого элемента (с проверкой на целочисленность размерности), после чего удаляем пустой элемент и константу после него из ОПС.

Программа **10**:

1. Переключаемся на таблицу целых чисел.

Программа **11**:

1. Переключаемся на таблицу вещественных чисел.

# **ФОРМАТ ОПС**

ОПС в программе будет представлена в виде std::vector<literal>

Возможные значения в ОПС: ссылка на переменную, операция, метка, константа.

enum typeLiteral : short {

link,

operation,

mark,

constant,

};

struct literal {

typeLiteral type;

any value;

};