

Вариант 1

Для грамматики $G[E]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[E]$:

1. $E \rightarrow TA$
2. $A \rightarrow \varepsilon \mid + TA \mid - TA$
3. $T \rightarrow OB$
4. $B \rightarrow \varepsilon \mid *OB \mid /OB$
5. $O \rightarrow \text{num} \mid \text{id} \mid (E)$

num – числовая константа $\mathbb{C}\{\mathbb{C}\}$, id – идентификатор $\mathbb{B}\{\mathbb{B}|\mathbb{C}\}$, $\mathbb{B} = \{a, b, c, \dots, z, A, B, \dots, Z\}$, $\mathbb{C} = \{0, 1, \dots, 9\}$.

Вариант 2

Для грамматики $G[\langle \text{условный оператор} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{условный оператор} \rangle]$:

1. $\langle \text{условный оператор} \rangle ::= \text{IF } \langle \text{условие} \rangle \text{ THEN } \langle \text{оператор} \rangle$
2. $\langle \text{условие} \rangle ::= \langle \text{выражение} \rangle \langle \text{операция отношения} \rangle$
 $\langle \text{выражение} \rangle$
3. $\langle \text{выражение} \rangle ::= \langle \text{терм} \rangle \{+ \langle \text{терм} \rangle\}$
4. $\langle \text{терм} \rangle ::= \langle \text{множитель} \rangle \{ * \langle \text{множитель} \rangle \}$
5. $\langle \text{множитель} \rangle ::= \langle \text{идентификатор} \rangle \mid (\langle \text{выражение} \rangle)$
6. $\langle \text{идентификатор} \rangle ::= \langle \text{буква} \rangle \{ \langle \text{буква} \rangle \mid \langle \text{число} \rangle \}$
7. $\langle \text{оператор} \rangle ::= \langle \text{выражение} \rangle \mid \langle \text{условный оператор} \rangle$
 $\langle \text{операция отношения} \rangle ::= \text{""} = \text{""} \mid \text{""} < \text{""} \mid \text{""} < = \text{""} \mid \text{""} > \text{""} \mid \text{""} > = \text{""} \mid \text{""} ! = \text{""}$
 $\langle \text{буква} \rangle ::= a \mid b \mid c \mid \dots \mid z \mid A \mid B \mid \dots \mid Z$
 $\langle \text{число} \rangle ::= \langle \text{цифра} \rangle \{ \langle \text{цифра} \rangle \}$
 $\langle \text{цифра} \rangle ::= 0 \mid 1 \mid \dots \mid 8 \mid 9$

Вариант 3

Для грамматики $G[E]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[E]$:

1. $E \rightarrow E + T \mid E - T \mid T$
2. $T \rightarrow T * F \mid T / F \mid F$
3. $F \rightarrow V \wedge F \mid V$
4. $V \rightarrow (E) \mid id \mid number \mid \epsilon$

number – числовая константа $\mathbb{C}\{\mathbb{C}\}$

id – идентификатор $B\{B|\mathbb{C}\}$

$B = \{a, b, c, \dots, z, A, B, \dots, Z\}$, $\mathbb{C} = \{0, 1, \dots, 9\}$

Вариант 4

Для грамматики $G[S]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[S]$:

1. $S \rightarrow \text{select } X \text{ from } X$
2. $X \rightarrow AY$
3. $Y \rightarrow ,AY \mid \epsilon$
4. $A \rightarrow \text{letter } \{\text{letter}\}$

letter $\rightarrow \{a, b, c, \dots, z, A, B, \dots, Z\}$

Примечание: Данная грамматика для упрощенного варианта SQLзапроса «select»: «select столбец1, столбец2, ... from таблица1, таблица2, ...»

Вариант 5

Для грамматики $G[<E>]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[<E>]$:

1. $E \rightarrow E1 \mid E1 [= \mid < \mid >] E1$
2. $E1 \rightarrow T \{ [+ \mid - \mid \text{“or”}] T \}$
3. $T \rightarrow F \{ [* \mid / \mid \text{“and”}] F \}$
4. $F \rightarrow \text{“true”} \mid \text{“false”} \mid \text{“not” } F \mid (E)$

Вариант 6

Для грамматики $G[<\text{Выражение}>]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[<\text{Выражение}>]$:

1. $<\text{Выражение}> = <\text{Слагаемое}> \{ (+|-) <\text{Слагаемое}> \}$
2. $<\text{Слагаемое}> = <\text{Множитель}> \{ (*|/) <\text{Множитель}> \}$
3. $<\text{Множитель}> = [+|-] <\text{ДробноеЧисло}> \mid <\text{Функция}>$
 $\mid (<\text{Выражение}>)$
4. $<\text{ДробноеЧисло}> = <\text{ЦелаяЧасть}> \mid <\text{ЦелаяЧасть}>$
 $.<\text{ДробнаяЧасть}>$
5. $<\text{ЦелаяЧасть}> = <\text{Цифра}> \{ <\text{Цифра}> \}$
6. $<\text{ДробнаяЧасть}> = <\text{Цифра}> \{ <\text{Цифра}> \}$
7. $<\text{Функция}> = <\text{ИмяФункции}> (<\text{Выражение}>)$
 $<\text{ИмяФункции}> = \text{“sin”} \mid \text{“cos”} \mid \text{“tg”} \mid \text{“ctg”} \mid \text{“log”} \mid \text{“ln”}$
 $<\text{Цифра}> = 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Вариант 7

Для грамматики $G[\langle \text{expr} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{expr} \rangle]$:

1. $\langle \text{expr} \rangle ::= \langle \text{expr_add_sub} \rangle \mid \langle \text{expr} \rangle \langle \text{operator_mul_div} \rangle \langle \text{expr_add_sub} \rangle$
2. $\langle \text{expr_add_sub} \rangle ::= \langle \text{term} \rangle \mid \langle \text{expr_add_sub} \rangle \langle \text{operator_mul_div} \rangle \langle \text{term} \rangle$
3. $\langle \text{term} \rangle ::= \langle \text{integer} \rangle \mid "(" \langle \text{expr} \rangle ")"$
4. $\langle \text{operator_mul_div} \rangle ::= "*" \mid "/" \mid "\%"$
5. $\langle \text{operator_add_sub} \rangle ::= "+" \mid "-"$
6. $\langle \text{integer} \rangle ::= \langle \text{digit} \rangle \langle \text{integer} \rangle \langle \text{digit} \rangle$
 $\langle \text{digit} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Вариант 8

Для грамматики $G[\langle \text{число} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{число} \rangle]$:

1. $\langle \text{число} \rangle ::= [\langle \text{знак} \rangle] \langle \text{целая часть} \rangle . [\langle \text{дробная часть} \rangle] [E [\langle \text{знак} \rangle] \langle \text{порядок} \rangle] \mid [\langle \text{знак} \rangle] [\langle \text{целая часть} \rangle] . \langle \text{дробная часть} \rangle [E [\langle \text{знак} \rangle] \langle \text{порядок} \rangle]$
2. $\langle \text{целая часть} \rangle ::= \langle \text{цифра} \rangle [\langle \text{целая часть} \rangle]$
3. $\langle \text{дробная часть} \rangle ::= \langle \text{целая часть} \rangle$
4. $\langle \text{порядок} \rangle ::= \langle \text{целая часть} \rangle$
 $\langle \text{цифра} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
 $\langle \text{знак} \rangle ::= + \mid -$

Вариант 9

Для грамматики $G[\langle \text{Выражение} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{Выражение} \rangle]$:

1. $\langle \text{Выражение} \rangle = \langle \text{Простое выражение} \rangle [\langle \text{Операция отношения} \rangle \langle \text{Простое выражение} \rangle]$
 2. $\langle \text{Простое выражение} \rangle = \langle \text{Терм} \rangle \{ \langle \text{Аддитивная операция} \rangle \langle \text{Терм} \rangle \}$
 3. $\langle \text{Терм} \rangle = \langle \text{Фактор} \rangle \{ \langle \text{Мультипликативная операция} \rangle \langle \text{Фактор} \rangle \}$
 4. $\langle \text{Фактор} \rangle = \langle \text{Переменная} \rangle \mid \text{"not"} \langle \text{Фактор} \rangle \mid \text{"("} \langle \text{Выражение} \rangle \text{"}"}$
 5. $\langle \text{Переменная} \rangle = \text{Буква} \{ \text{Буква} \mid \text{Цифра} \}$
 6. $\langle \text{Операция отношения} \rangle = \text{"="} \mid \text{"!="} \mid \text{"<"}$ $\mid \text{"<="} \mid \text{">"}$ $\mid \text{">="}$
 7. $\langle \text{Аддитивная операция} \rangle = \text{"+"} \mid \text{"-"} \mid \text{"or"}$
 8. $\langle \text{Мультипликативная операция} \rangle = \text{"*"} \mid \text{"/"}$ $\mid \text{"\%"} \mid \text{"^"}$
- Буква – [a, b, c, ..., z, A, B, ..., Z], Цифра – [0, 1, ..., 9]

Вариант 10

Для грамматики $G[\langle \text{ФОРМУЛА} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{ФОРМУЛА} \rangle]$:

1. $\langle \text{ФОРМУЛА} \rangle \rightarrow \langle \text{ФОРМУЛА} \rangle \langle \text{ЗНАК} \rangle \langle \text{ФОРМУЛА} \rangle \mid \langle \text{ЧИСЛО} \rangle \mid \text{"("} \langle \text{ФОРМУЛА} \rangle \text{"}"}$
2. $\langle \text{ЧИСЛО} \rangle \rightarrow \langle \text{ЦИФРА} \rangle \mid \langle \text{ЧИСЛО} \rangle \langle \text{ЦИФРА} \rangle$
3. $\langle \text{ЗНАК} \rangle \rightarrow \text{"+"} \mid \text{"-"} \mid \text{"*"} \mid \text{"/"}$
4. $\langle \text{ЦИФРА} \rangle \rightarrow \text{"0"} \mid \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"}$

Вариант 11

Для грамматики $G[S]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[S]$:

1. $S \rightarrow NP VP$
2. $NP \rightarrow Pro \mid PropN \mid Det A Nom \mid Det Nom$
3. $Nom \rightarrow Nom PP \mid N Nom \mid N$
4. $VP \rightarrow VP PP \mid V NP \mid V NP PP \mid V PP$
5. $PP \rightarrow P NP$

$N \rightarrow flight \mid passenger \mid trip \mid morning \mid \dots$

$V \rightarrow is \mid prefers \mid like \mid need \mid depend \mid fly \mid \dots$

$A \rightarrow cheapest \mid non-stop \mid first \mid latest other \mid direct \mid \dots$

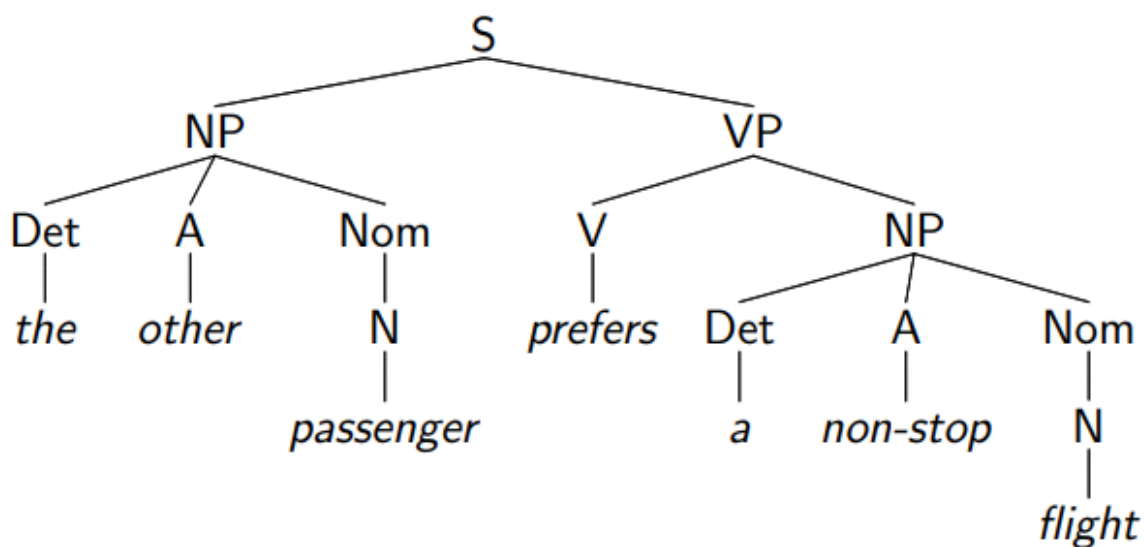
$Pro \rightarrow me \mid I \mid you \mid it \mid \dots$

$PropN \rightarrow Alaska \mid Baltimore \mid Los Angeles \mid Chicago \mid \dots$

$Det \rightarrow the \mid a \mid an \mid this \mid these \mid that \mid \dots$

$P \rightarrow from \mid to \mid on \mid near \mid \dots$

Примечание: грамматика реализует составление простейших предложений на английском языке.



Вариант 12

Для грамматики $G[E]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[E]$:

1. $E \rightarrow E \Rightarrow E_1 \mid E_1$
2. $E_1 \rightarrow E_1 \vee E_2 \mid E_2$
3. $E_2 \rightarrow E_2 \wedge E_3 \mid E_3$
4. $E_3 \rightarrow \neg E_4 \mid E_4$
5. $E_4 \rightarrow (E) \mid \text{id}$

id – идентификатор $B\{B|C\}$

$B = \{a, b, c, \dots, z, A, B, \dots, Z\}$, $C = \{0, 1, \dots, 9\}$

Примечание: данными productions описывается язык булевой логики (в окне ввода текста замените символы эквивалентными по смыслу).

Вариант 13

Для грамматики $G[HTML]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[HTML]$:

1. $HTML ::= CONTENT \mid TAG \ HTML \ TAG$
2. $TAG ::= \langle \text{html} \rangle \mid \langle / \text{html} \rangle \mid \langle \text{head} \rangle \mid \langle / \text{head} \rangle \mid \langle \text{title} \rangle \mid \langle / \text{title} \rangle \mid \langle \text{body} \rangle \mid \langle / \text{body} \rangle \mid \langle \text{h2} \rangle \mid \langle / \text{h2} \rangle \mid \langle \text{p} \rangle \mid \langle / \text{p} \rangle$
3. $CONTENT ::= \text{text}$

text – $B\{B\}$

$B = \{a, b, c, \dots, z, A, B, \dots, Z\}$

Примечание: данными productions описывается язык HTML-разметки.

Вариант 14

Для грамматики $G[STMT]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[STMT]$:

1. $STMT ::= \text{if "(" EXPR "(") STMT} \mid \text{if "(" EXPR "(") STMT else STMT}$

2. $EXPR ::= \text{VARIABLE OPERATOR VALUE}$

$VALUE$ – числовая константа $\mathbb{C}\{\mathbb{C}\}$, $VARIABLE$ – переменная $B\{B|\mathbb{C}\}$, $B = \{a, b, c, \dots, z, A, B, \dots, Z\}$, $\mathbb{C} = \{0, 1, \dots, 9\}$, $OPERATOR = "=" \mid "<" \mid "<=" \mid ">" \mid ">=" \mid "!="$

Вариант 15

Для грамматики $G[Doc]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[Doc]$:

1. $Doc \rightarrow \lambda \mid \text{Element Doc}$

2. $\text{Element} \rightarrow \text{Text} \mid \text{ Doc } \mid \text{<p> Doc </p>} \mid \text{ List }$

3. $List \rightarrow \lambda \mid \text{ListItem List}$

4. $\text{ListItem} \rightarrow \text{ Text }$

5. $\text{Text} \rightarrow \lambda \mid \text{Char Text}$

$\text{Char} \rightarrow a \mid b \mid c \mid \dots \mid z \mid A \mid B \mid \dots \mid Z$

Примечание: данными productions описывается язык HTML-разметки.

Вариант 16

Для грамматики $G[\text{Program}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{Program}]$:

1. $\text{Program} \rightarrow \varepsilon \mid \text{Instr Program}$
2. $\text{Instr} \rightarrow '+' \mid '-' \mid '>' \mid '<' \mid ',' \mid '.' \mid '[' \text{Program} ']'$

Примечание: данными productions описывается эзотерический язык программирования Brainfuck.

Вариант 17

Для грамматики $G[\text{Program}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{Program}]$:

1. $\text{Program} \rightarrow \varepsilon \mid \text{Instr Program}$
2. $\text{Instr} \rightarrow '+' \mid '-' \mid '*' \mid '/' \mid '_' \mid '=' \mid '>' \mid '&' \mid '|' \mid '~' \mid '$' \mid '\%' \mid '\backslash' \mid '@' \mid '[' \text{Program} ']'$

Примечание: данными productions описывается эзотерический язык программирования FALSE.

Вариант 18

Для грамматики $G[S]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[S]$:

1. $S \rightarrow \langle \text{Noun phrase} \rangle \langle \text{Verb phrase} \rangle$
 2. $\langle \text{Noun phrase} \rangle \rightarrow \langle \text{Noun} \rangle | \langle \text{Adjective phrase} \rangle \langle \text{Noun} \rangle | \lambda$
 3. $\langle \text{Verb phrase} \rangle \rightarrow \langle \text{Verb} \rangle \langle \text{Noun phrase} \rangle$
 4. $\langle \text{Adjective phrase} \rangle \rightarrow \langle \text{Adjective phrase} \rangle \langle \text{Adjective} \rangle | \lambda$
- $\langle \text{Noun} \rangle \rightarrow \text{flight} | \text{passenger} | \text{trip} | \text{morning} | \dots$
 $\langle \text{Verb} \rangle \rightarrow \text{is} | \text{prefers} | \text{like} | \text{need} | \text{depend} | \text{fly} | \dots$
 $\langle \text{Adjective} \rangle \rightarrow \text{non-stop} | \text{first} | \text{direct} | \dots$

Вариант 19

Для грамматики $G[\text{stmt}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{stmt}]$:

1. $\text{stmt} \rightarrow \text{IF exp stmt} | \text{IF exp stmt ELSE stmt} | \text{ID ASSIGN exp SEMICOLON}$
2. $\text{exp} \rightarrow \text{TRUE} | \text{FALSE} | \text{exp OR exp} | \text{exp AND exp} | \text{NOT exp} | \text{exp}$

ID – переменная $B\{B|C\}$, $B = \{a, b, c, \dots, z, A, B, \dots, Z\}$, $C = \{0, 1, \dots, 9\}$, ASSIGN – “==” | “<” | “<=” | “>” | “>=” | “!=”

Вариант 20

Для грамматики $G[\text{begin-stmt}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{begin-stmt}]$:

1. $\text{begin-stmt} \rightarrow \text{begin stmt-list end}$
2. $\text{stmt-list} \rightarrow \text{stmt} \mid \text{stmt} ; \text{stmt-list}$
3. $\text{stmt} \rightarrow \text{if-stmt} \mid \text{while-stmt} \mid \text{begin-stmt} \mid \text{assg-stmt}$
4. $\text{if-stmt} \rightarrow \text{if bool-expr then stmt else stmt}$
5. $\text{while-stmt} \rightarrow \text{while bool-expr do stmt}$
6. $\text{assg-stmt} \rightarrow \text{VAR} := \text{arith-expr}$
7. $\text{bool-expr} \rightarrow \text{arith-expr compare-op arith-expr}$
8. $\text{arith-expr} \rightarrow \text{VAR} \mid \text{NUM} \mid (\text{arith-expr}) \mid \text{arith-expr} + \text{arith-expr} \mid \text{arith-expr} * \text{arith-expr}$

VAR – переменная $B\{B|C\}$, $B = \{a, b, c, \dots, z, A, B, \dots, Z\}$, NUM – $\{0, 1, \dots, 9\}$, compare-op – $"==" \mid "<" \mid "<=" \mid ">" \mid ">=" \mid "!="$

Вариант 21

Для грамматики $G[\text{<variable-declaration>}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

1. $\text{<variable-declaration>} ::= \text{var } \text{<integer-constant>} \{ ; \text{<variable-declaration>} \}$
2. $\text{<integer-constant>} ::= [+ \mid -] \text{<digit>} \{ \text{<digit>} \}$

Вариант 22

Для грамматики $G[\text{expression}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{expression}]$:

1. $\text{expression} := \text{multExpression} \mid \text{powExpression} \mid \text{expression} ('+'|-') \text{multExpression} \mid \text{expression} ('+'|-') \text{powExpression}$

2. $\text{multExpression} := \text{ID} \mid \text{multExpression} ('*'|'/') \text{ID}$

3. $\text{powExpression} := \text{ID} '^' \text{powExpression} \mid \text{ID}$

ID – идентификатор $\mathcal{B}\{\mathcal{B}|\mathcal{C}\}$, $\mathcal{B} = \{a, b, c, \dots, z, A, B, \dots, Z\}$, NUM – $\{0, 1, \dots, 9\}$

Вариант 23

Для грамматики $G[\text{begin-stmt}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{begin-stmt}]$:

1. $\text{begin-stmt} \rightarrow \text{begin stmt-list end}$

2. $\text{stmt-list} \rightarrow \text{stmt} \mid \text{stmt} ; \text{stmt-list}$

3. $\text{stmt} \rightarrow \text{begin-stmt} \mid \text{assg-stmt}$

4. $\text{assg-stmt} \rightarrow \text{VAR} := \text{arith-expr}$

5. $\text{arith-expr} \rightarrow \text{VAR} \mid \text{NUM} \mid (\text{arith-expr}) \mid \text{arith-expr} + \text{arith-expr} \mid \text{arith-expr} * \text{arith-expr}$

VAR – переменная $\mathcal{B}\{\mathcal{B}|\mathcal{C}\}$, $\mathcal{B} = \{a, b, c, \dots, z, A, B, \dots, Z\}$, NUM – $\{0, 1, \dots, 9\}$, compare-op – $"==" \mid "<" \mid "<=" \mid ">" \mid ">=" \mid "!="$

Вариант 24

Для грамматики $G[\text{unsigned_number}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\text{unsigned_number}]$:

1. $\text{unsigned_number} ::= \text{decimal_number} \mid \text{exponential_part} \mid \text{decimal_number} \text{exponential_part}$
2. $\text{decimal_number} ::= \text{unsigned_integer} \mid \text{decimal_fraction} \mid \text{unsigned_integer} \text{decimal_fraction}$
3. $\text{unsigned_integer} ::= \text{digit} \mid \text{unsigned_integer} \text{digit}$
4. $\text{integer} ::= \text{unsigned_integer} \mid '+' \text{unsigned_integer} \mid '-' \text{unsigned_integer}$
5. $\text{decimal_fraction} ::= '.' \text{unsigned_integer}$
6. $\text{exponential_part} ::= '10' \text{integer}$
- $\text{digit} ::= 0 \mid 1 \mid \dots \mid 9$

Примечание: данные productions были взяты из грамматики языка ALGOL 60.

Вариант 25

Для грамматики $G[\langle \text{While} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{While} \rangle]$:

1. $\langle \text{While} \rangle \rightarrow \text{while } \langle \text{Cond} \rangle \text{ do } \langle \text{Stmt} \rangle ;$
2. $\langle \text{Cond} \rangle \rightarrow \langle \text{LogExpr} \rangle \{ \text{or } \langle \text{LogExpr} \rangle \}$
3. $\langle \text{LogExpr} \rangle \rightarrow \langle \text{RelExpr} \rangle \{ \text{and } \langle \text{RelExpr} \rangle \}$
4. $\langle \text{RelExpr} \rangle \rightarrow \langle \text{Operand} \rangle [\text{rel } \langle \text{Operand} \rangle]$
5. $\langle \text{Operand} \rangle \rightarrow \text{var} \mid \text{const}$
6. $\langle \text{Stmt} \rangle \rightarrow \text{var as } \langle \text{ArithExpr} \rangle$
7. $\langle \text{ArithExpr} \rangle \rightarrow \langle \text{Operand} \rangle \{ \text{ao } \langle \text{Operand} \rangle \}$

Примечание: while, do, and, or – ключевые слова. В тип rel объединили операции сравнения $<$, $<=$, $>=$, $>$, $!=$ и $==$, в тип ao арифметические операции $+$ и $-$, в тип as оператор присваивания $=$, тип var – название переменной (только буквы), тип const – числа. Причина, по которой не объединены в один тип логические операции and и or заключается в том, что эти операции имеют различный приоритет. Пример цепочки: while a < b and b <= c do b=b+c-20 end ;

Вариант 26

Для грамматики $G[\langle \text{While} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{While} \rangle]$:

1. $\langle \text{While} \rangle \rightarrow \text{do } \langle \text{Stmt} \rangle \text{ while } \langle \text{Cond} \rangle ;$
2. $\langle \text{Cond} \rangle \rightarrow \langle \text{LogExpr} \rangle \{ \text{or } \langle \text{LogExpr} \rangle \}$
3. $\langle \text{LogExpr} \rangle \rightarrow \langle \text{RelExpr} \rangle \{ \text{and } \langle \text{RelExpr} \rangle \}$
4. $\langle \text{RelExpr} \rangle \rightarrow \langle \text{Operand} \rangle [\text{rel } \langle \text{Operand} \rangle]$
5. $\langle \text{Operand} \rangle \rightarrow \text{var} \mid \text{const}$
6. $\langle \text{Stmt} \rangle \rightarrow \text{var as } \langle \text{ArithExpr} \rangle$
7. $\langle \text{ArithExpr} \rangle \rightarrow \langle \text{Operand} \rangle \{ \text{ao } \langle \text{Operand} \rangle \}$

Примечание: while, do, and, or – ключевые слова. В тип rel объединили операции сравнения $<$, $<=$, $>=$, $>$, $!=$ и $==$, в тип ao арифметические операции $+$ и $-$, в тип as оператор присваивания $=$, тип var – название переменной (только буквы), тип const – числа. Причина, по которой не объединены в один тип логические операции and и or заключается в том, что эти операции имеют различный приоритет. Пример цепочки: while a < b and b <= c do b=b+c-20 ;

Вариант 27

Для грамматики $G[\langle \text{For} \rangle]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

$G[\langle \text{For} \rangle]$:

1. $\langle \text{For} \rangle \rightarrow \text{for id} := \langle \text{Operand} \rangle \text{ to } \langle \text{Operand} \rangle \text{ do } \langle \text{Stmt} \rangle ;$
2. $\langle \text{Operand} \rangle \rightarrow \text{var} \mid \text{const}$
3. $\langle \text{Stmt} \rangle \rightarrow \text{var as } \langle \text{ArithExpr} \rangle$
4. $\langle \text{ArithExpr} \rangle \rightarrow \langle \text{Operand} \rangle \{ \text{ao } \langle \text{Operand} \rangle \}$

Примечание: for, do, and, or – ключевые слова. В тип ao объединили арифметические операции + и -, в тип as оператор присваивания =, тип var – название переменной (только буквы), тип var – числа, тип id – идентификаторы (буквы и цифры). Причина, по которой не объединены в один тип логические операции and и or заключается в том, что эти операции имеют различный приоритет. Пример цепочки: for i3:=10 to n do x=y-z+70 ;

Вариант 28

Для грамматики $G[S]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

1. $S \rightarrow Q V \mid V Q$
2. $Q \rightarrow N_{\text{им}}$
3. $V \rightarrow V O \mid O V \mid V$
4. $N_z \rightarrow A_z N_z \mid N_z \text{ // } z - \text{один из шести падежей русского языка}$
5. $V \rightarrow \text{шла} \mid \text{бежала} \mid \text{летела} \mid \text{скакала} \mid \dots$
6. $N_z \rightarrow \text{деревняз} \mid \text{дорогаз} \mid \text{студентказ} \mid \text{ведьмаз} \mid \text{метлаз} \dots$
7. $A_z \rightarrow \text{красиваяз} \mid \text{молодаяз} \mid \text{широкаяз} \mid \text{пыльнаяз} \mid \text{ночнаяз} \dots$
8. $O \rightarrow \text{пешком} \mid \text{верхом} \mid \text{галопом} \mid \text{на } N_{\text{предл}} \mid \text{над } N_{\text{твор}} \mid \text{в } N_{\text{предл}} \mid \text{по } N_{\text{дат}} \dots$

Примечание: данные продукции реализуют грамматику фрагмента русского языка.

Вариант 29

Для грамматики $G[\text{Lexp-seq}]$ разработать и реализовать алгоритм анализа на основе метода рекурсивного спуска.

1. $\text{Lexp-seq} ::= \text{Lexp-seq Lexp} \mid \text{Lexp}$
2. $\text{Lexp} ::= \text{Atom} \mid \text{List}$
3. $\text{Atom} ::= \text{„number“} \mid \text{„identifier“}$
4. $\text{List} ::= \text{„(“ Lexp-seq „)“}$

Примечание: данная грамматика порождает цепочки упрощенного языка Lisp.