Тут будет титульник

ОГЛАВЛЕНИЕ

ПОСТАНОВКА ЗАДАЧИ

Разработать распознаватель модельного языка программирования, выполнив следующие действия.

1. В соответствии с номером варианта составить формальное описание модельного языка программирования с помощью:
2. РБНФ;
3. диаграмм Вирта;
4. формальных грамматик.
5. Написать пять содержательных примеров программ, раскрывающих особенности конструкций модельного языка программирования, отразив в этих примерах все его функциональные возможности.
6. Составить таблицы лексем и диаграмму состояний с действиями для распознавания и формирования лексем языка.
7. По диаграмме с действиями написать функцию сканирования текста входной программы на модельном языке.
8. Разработать программное средство, реализующее лексический анализ текста программы на входном языке.
9. Реализовать синтаксический анализатор текста программы на модельном языке методом рекурсивного спуска.
10. Построить цепочку вывода и дерево разбора простейшей программы на модельном языке из начального символа грамматики.
11. Дополнить синтаксический анализатор процедурами проверки семантической правильности программы на модельном языке в соответствии с контекстными условиями вашего варианта.
12. Распечатать пример таблиц идентификаторов и двуместных операций.
13. Показать динамику изменения содержимого стека при семантическом анализе программы на примере одного синтаксически правильного выражения.
14. Составить набор контрольных примеров, демонстрирующих все возможные типы лексических, синтаксических и семантических ошибок в программах на модельном языке.

ФОРМАЛЬНАЯ МОДЕЛЬ ЗАДАЧИ

1. **РБНФ**

<программа>::= program var <описание> begin <оператор> {; <оператор>} end.

<описание>::= {<идентификатор> {, <идентификатор> } : <тип> ;}

<тип>::= % | ! | $

<оператор>::= <составной> | <присваивания> | <условный> | <фиксированного\_цикла> | <условного\_цикла> | <ввода> | <вывода>

<составной>::= «[» <оператор> { : <оператор> } «]»

<присваивания>::= <идентификатор> as <выражение>

<условный>::= if <выражение> then <оператор> [ else <оператор>]

<фиксированного\_цикла>::= for <присваивания> to <выражение> do <оператор>

<условного\_цикла>::= while <выражение> do <оператор>

<ввода>::= read «(»<идентификатор> {, <идентификатор> } «)»

<вывода>::= write «(»<выражение> {, <выражение> } «)»

<операции\_группы\_отношения>:: = < > | = | < | <= | > | >=

<операции\_группы\_сложения>:: = + | - | or

<операции\_группы\_умножения>::= \* | / | and

<унарная\_операция>::= not

<выражение>::= <операнд>{<операции\_группы\_отношения> <операнд>}

<операнд>::= <слагаемое> {<операции\_группы\_сложения> <слагаемое>}

<слагаемое>::= <множитель> {<операции\_группы\_умножения> <множитель>}

<множитель>::= <идентификатор> | <число> | <логическая\_константа> | <унарная\_операция> <множитель> | «(»<выражение>«)»

<число>::= <целое> | <действительное>

<логическая\_константа>::= true | false

<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

<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)

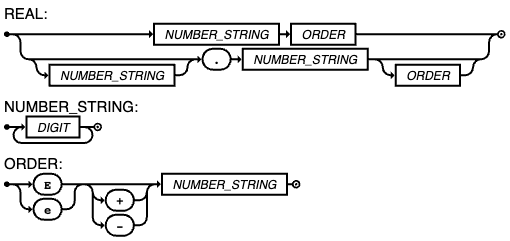
<действительное>::= <числовая\_строка> <порядок> | [<числовая\_строка>] .<числовая\_строка> [порядок]

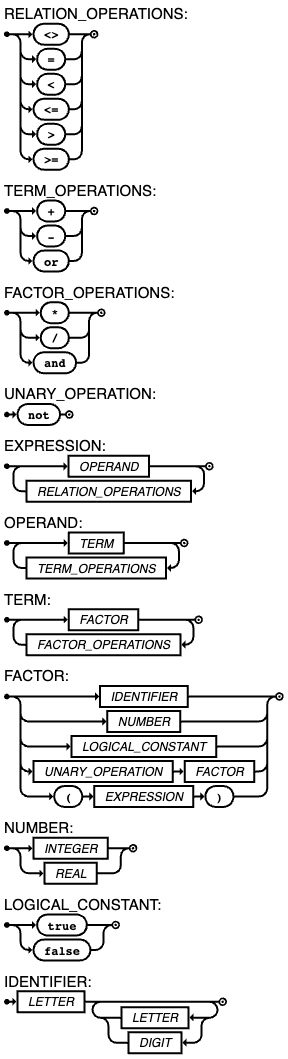
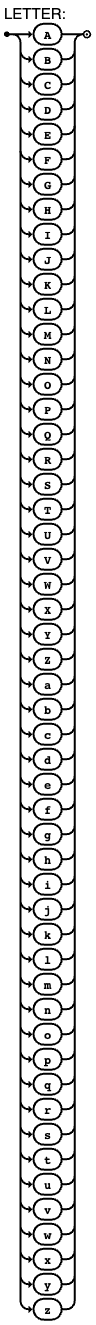
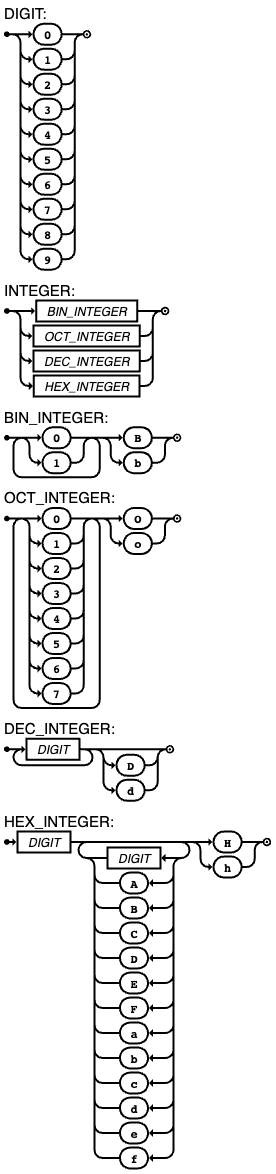
<числовая\_строка>::= {/ <цифра> /}

<порядок>::= ( E | e )[+ | -] <числовая\_строка>

1. **Диаграмма вирта**





1. **Формальные грамматики**

PROGRAM → program var DESCRIPTION begin OPERATOR1

OPERATOR1 → OPERATOR ; OPERATOR1| OPERATOR

DESCRIPTION → IDENTIFIER1 : TYPE ; DESCRIPTION| IDENTIFIER1 : TYPE;

IDENTIFIER1 → IDENTIFIER1, IDENTIFIER | IDENTIFIER

TYPE → % | ! | $

OPERATOR → COMPOSITE\_OPERATOR | ASSIGNMENT\_OPERATOR | CONDITIONAL\_OPERATOR | FIXED\_CYCLE\_OPERATOR | CONDITIONAL\_CYCLE\_OPERATOR | INPUT\_OPERATOR | OUTPUT\_OPERATOR

COMPOSITE\_OPERATOR → [COMPOSITE\_OPERATOR1]

COMPOSITE\_OPERATOR1 → OPERATOR | COMPOSITE\_OPERATOR1 : OPERATOR

ASSIGNMENT\_OPERATOR → IDENTIFIER as EXPRESSION

CONDITIONAL\_OPERATOR → if EXPRESSION then OPERATOR | if EXPRESSION then OPERATOR else OPERATOR

FIXED\_CYCLE\_OPERATOR → for ASSIGNMENT\_OPERATOR to EXPRESSION do OPERATOR

CONDITIONAL\_CYCLE\_OPERATOR → while EXPRESSION do OPERATOR

INPUT\_OPERATOR → read (INPUT\_OPERATOR1)

INPUT\_OPERATOR1 → INPUT\_OPERATOR1 , IDENTIFIER | IDENTIFIER

OUTPUT\_OPERATOR → write ( OUTPUT\_OPERATOR1)

OUTPUT\_OPERATOR1 → OUTPUT\_OPERATOR1, EXPRESSION | EXPRESSION

RELATION\_OPERATIONS → <> | = | < | <= | > | >=

TERM\_OPERATIONS → + | - | or

FACTOR\_OPERATIONS → \* | / | and

UNARY\_OPERATION → not

EXPRESSION → OPERAND | OPERAND EXPRESSION1

EXPRESSION1 → RELATION\_OPERATIONS OPERAND | RELATION\_OPERATIONS OPERAND EXPRESSION1

OPERAND → TERM | TERM OPERAND1

OPERAND1 → TERM\_OPERATIONS TERM | OPERAND1 TERM\_OPERATIONS TERM

TERM → FACTOR | FACTOR TERM1

TERM1 → FACTOR\_OPERATIONS FACTOR | TERM1 FACTOR\_OPERATIONS FACTOR

FACTOR → IDENTIFIER | NUMBER | LOGICAL\_CONSTANT | UNARY\_OPERATION FACTOR | ( EXPRESSION )

NUMBER → INTEGER | REAL

LOGICAL\_CONSTANT → true | false

IDENTIFIER → IDENTIFIERLETTER | IDENTIFIERDIGIT | LETTER

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

INTEGER → BIN\_INTEGER | OCT\_INTEGER | DEC\_INTEGER | HEX\_INTEGER

BIN\_INTEGER → BIN\_INTEGER1 (B | b)

BIN\_INTEGER1 → 0 | 1 | BIN\_INTEGER1 1 | BIN\_INTEGER1 0

OCT\_INTEGER → OCT\_INTEGER1 O | OCT\_INTEGER1 o

OCT\_INTEGER1 → OCT\_INTEGER1 0 | OCT\_INTEGER1 1 | OCT\_INTEGER1 2 | OCT\_INTEGER1 3 | OCT\_INTEGER1 4 | OCT\_INTEGER1 5 | OCT\_INTEGER1 6 | OCT\_INTEGER1 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

DEC\_INTEGER → DEC\_INTEGER1 | DEC\_INTEGER1 D | DEC\_INTEGER1 d

DEC\_INTEGER1 → DIGIT | DEC\_INTEGER1 DIGIT

HEX\_INTEGER → HEX\_INTEGER1 H | HEX\_INTEGER1 h

HEX\_INTEGER1 → DIGIT | HEX\_INTEGER1 HEX\_INTEGER2

HEX\_INTEGER2 → DIGIT | A | B | C | D | E | F | a | b | c | d | e | f

REAL → NUMBER\_STRING ORDER | NUMBER\_STRING . NUMBER\_STRING ORDER |. NUMBER\_STRING ORDER | NUMBER\_STRING . NUMBER\_STRING | . NUMBER\_STRING

NUMBER\_STRING → DIGIT | NUMBER\_STRING DIGIT

ORDER → ORDER1 NUMBER\_STRING | ORDER1 + NUMBER\_STRING | ORDER1 - NUMBER\_STRING

ORDER1 → E | e