

Грамматика языка МИКРОЛИСП.

Содержательная расшифровка символов нетерминалов приведена в словаре SyntaxClasses21.rtf . Рекомендую держать этот файл перед глазами в отдельном окне.

```
# $mlisp21
  $id   $idq   $dec   $int
  $bool $str   (      )
  +     -     *      /
  <     =     >      <=
  >=    and   not     or
  cond  else  if      let
define display newline set!
```

#
NB!!! Ключевые слова, перечисленные в алфавите, записываются в цепочках как терминалы!

```
S -> PROG #1
PROG -> CALCS #2 |
      DEFS #3 |
      DEFS CALCS #4
```

Продукция #4 формирует программу как перечень определений, за которым следует перечень вычислений. Записать определения вперемешку с вычислениями нельзя – сначала все определения, затем все вычисления.

Продукции #2 и #3 позволяют исключить один из этих перечней.

```
E -> $id #5 |
     $int #6 |
     $dec #7 |
     AREX #8 |
     COND #9 |
     IF #10 |
     EASYLET #11 |
     CPROC #12
```

Синтаксический класс E (числовое выражение) объединяет большой набор разных подклассов.

В него входят числовые переменные и константы(токен \$id), числовые литералы(токены \$int и \$dec), арифметические выражения(класс AREX), числовой cond (класс COND), числовой if (класс IF), легкий let (класс EASYLET), вызов процедуры (класс CPROC).

Все выражения класса E вырабатывают числовые значения.

CPROC -> HCPROC) #13

HCPROC -> (\$id #14 |

HCPROC E #15

Продукция #13 завершает цепочку вызова числовой процедура токеном).

Продукция #14 вставляет в начало цепочки имя процедуры.

Продукция #15 добавляет аргументы. Все аргументы – числовые выражения.

Если продукция #15 не применяется, то получится цепочка (\$id) для вызова процедуры без параметров.

AREX -> HAREX E) #16

HAREX -> (AROP #17 |

HAREX E #18

Продукция #16 завершает выражение и вставляет в него последний операнд.

Продукция \$18 добавляет дополнительные операнды. Все операнды – числовые выражения. Если продукция #18 не променялась, то получится одноместное выражение.

Цепочки вида (+) не порождаются.

AROP -> + #19 |

- #20 |

*** #21 |**

/ #22

Синтаксический класс AROP объединяет знаки арифметических операций.

IF -> (if BOOL E E) #23

В числовом if и следствие, и альтернатива – числовые выражения.

EASYLET -> HEASYL E) #24

HEASYL -> (let () #25 |

HEASYL INTER #26

Продукция #24 завершает легкий let числовым выражением, которое вычисляет его значение.

Каждое применение продукции #26 вставляет внутреннее выражение класса INTER(см. ниже).

COND -> (cond BRANCHES) #27

BRANCHES -> ELSE #28 |

CLAUS #29 |

CLAUS BRANCHES #30

CLAUS -> (BOOL CLAUSB) #31

CLAUSB -> E #32 |

INTER CLAUSB #33

ELSE -> (else ELSEB) #34

ELSEB -> E #35 |

INTER ELSEB #36

Продукция #27 формирует числовой cond, вставляя в него список ветвей.

Продукция #28 создает список ветвей, помещая в него заключительную ветвь else, а продукция #27 создает список с заключительной клаузой. В посленем случае список вообще не содержит ветви else и состоит из одних клауз.

Продукция #30 «накачивает» список дополнительными клаузами.

Обратите внимание на то, что праворекурсивная продукция #30 добавляет элемент в начало списка, в отличие от леворекурсивной продукции #26, которая добавляет элемент в конец списка.

Продукция #31 вставляет в начало клаузы булевское выражение-предикат.

Продукция #32 завершает клаузу числовым выражением, которое вычисляет значение ветви.

Каждое применение продукции #33 помещает в тело клаузы внутреннее выражение класса INTER(см. ниже).

Ветвь else формируется аналогично клаузе.

**STR -> \$str #37 |
SIF #38 |
SCOND #39**

Класс строковых выражений STR объединяет строковые литералы(токен \$str) и условные строковые выражения (классы SIF и SCOND).

Строковые выражения можно использовать только в перечне вычислений и в качестве аргумента оператора display.

**SIF -> (if BOOL STR STR) #40
SCOND -> (cond SBRANCHES) #41
SBRANCHES -> SELSE #42 |
SCLAUS SBRANCHES #43
SCLAUS -> (BOOL STR) #44
SELS -> (else STR) #45**

Строковый cond всегда оканчивается ветвью else.

**BOOL -> \$bool #46 |
\$idq #47 |
CPRED #48 |
REL #49 |
OR #50 |
AND #51 |
(not BOOL) #52**

Синтаксический класс BOOL объединяет все виды булевских выражений: литерал(токен \$bool), переменную(токен \$idq), вызов процедуры-предиката (класс CPRED), отношение(класс REL), дизъюнкцию (класс OR), конъюнкцию(класс AND) и отрицание.

**CPRED -> HCPRED) #53
HCPRED -> (\$idq #54 |
HCPRED ARG #55
ARG -> E #56 |
BOOL #57**

Вызов процедуры-предиката(класс CPRED) формируется аналогично вызову числовой процедуры (класс CPROC), но в отличие от него может иметь как числовые, так и булевские аргументы.

REL -> (= E E) #58 |
 (< E E) #59 |
 (> E E) #60 |
 (>= E E) #61 |
 (<= E E) #62
 OR -> HOR BOOL) #63
 HOR -> (or #64 |
 HOR BOOL #65
 AND -> HAND BOOL) #66
 HAND -> (and #67 |
 HAND BOOL #68

Дизъюнкции и конъюнкции имеют структуру, похожую на арифметическую композицию(см. класс AREX), но определены для булевских операндов.

NB!!! В МИКРОЛИСПе отсутствует неявное приведение числовых значений к булевым и наоборот.

Поэтому в контекстах, отмеченных нетерминалом BOOL, нельзя использовать числовые выражения, а в контекстах, отмеченных нетерминалом E, булевские.

SET -> HSET E) #69
 HSET -> (set! \$id #70

Продукция #70 определяет переменную, а продукция #69 – выражение, вычисляющее новое значение переменной.

DISPSET -> (display E) #71 |
 (display BOOL) #72 |
 (display STR) #73 |
 (newline) #74 |
 SET #75

Класс DISPSET объединяет операторы вывода и присваивания.

INTER -> DISPSET #76 |
 E #77

Класс INTER объединяет все выражения, которые можно записать внутри ветвей cond, в теле легкого let, блока и числовой процедуры.

CALCS -> CALC #78 |

CALCS CALC #79

Продукция #78 вставляет в перечень первое вычисление.

Продукция #79 наращивает перечень вычислениями.

```
CALC -> E #80 |  
      BOOL #81 |  
      STR #82 |  
      DISPSET #83
```

Вычисление – это выражение, записанное вне тела какой-либо процедуры.

```
DEFS -> DEF #84 |  
      DEFS DEF #85  
      DEF -> PRED #86 |  
      VAR #87 |  
      PROC #88
```

Класс DEF объединяет все виды определений, которые можно сделать с помощью формы define: процедур-предикатов, переменных и числовых процедур. Определение можно поместить только в начале программы вне тела процедуры (см. класс PROG). Поэтому в МИКРОЛИСПе нет вложенных процедур.

```
PRED -> HPRED BOOL ) #89  
HPRED -> PDPAR ) #90  
PDPAR -> ( define ( $idq #91 |  
      PDPAR $idq #92 |  
      PDPAR $id #93
```

Продукция #89 формирует конец тела процедуры-предиката и вставляет в него булеское выражение, вычисляющее значение процедуры.

Продукция #90 завершает список параметров процедуры-предиката.

Продукция #91 формирует начало определения процедуры-предиката с именем \$idq.

Продукция #92 добавляет в список параметров булевский идентификатор \$idq.

Продукция #93 добавляет в список параметров числовой идентификатор \$id.

```
VAR -> VARDCL E ) #94
```

VARDECL -> (define \$id #95

Продукция #95 формирует объявление переменной с именем \$id.

Продукция #94 добавляет инициализатор.

Эту форму можно использовать только вне процедур для определения глобальных переменных.

Для определения локальных переменных ее использовать нельзя.

PROC -> HPROC BLOCK) #96 |

HPROC E) #97

HPROC -> PCPAR) #98 |

HPROC INTER #99

PCPAR -> (define (\$id #100 |

PCPAR \$id #101

Продукция #96 формирует конец тела числовой процедуры и вставляет в него блок.

Альтернативная продукция #97 завершает тело процедуры числовым выражением.

Продукция #98 завершает список параметров.

Продукция #99 вставляет в тело процедуры дополнительные выражения класса INTER.

В отличие от процедуры-предиката тело числовой процедуры может быть составным.

Продукция #100 формирует начало определения числовой процедуры с именем \$id.

Продукция #101 добавляет в список параметров имя \$id. У числовой процедуры булевских параметров быть не может.

BLOCK -> HBLOCK E) #102

HBLOCK -> BLVAR) #103 |

HBLOCK INTER #104

BLVAR -> (let (LOCDEF #105 |

BLVAR LOCDEF #106

LOCDEF -> (\$id E) #107

Продукция #102 завершает тело блока числовым выражением.

Продукция #103 завершает список локальных переменных.

Продукция #104 вставляет в тело блока дополнительные выражения класса INTER.

Продукция #105 формирует начало блока и вставляет в него определение первой локальной переменной.

Продукция #106 наращивает список локальных переменных.

Продукция #107 формирует определение локальной переменной с именем \$id и числовым инициализатором.