

Давайте сконструируем язык **L** через **ПРЯМУЮ (префиксную) польскую запись**.

Зададим алфавит $\Sigma_0 = \{a, b, \dots, z, A, B, \dots, Z, 0, \dots, 9, _ \}$ - буквы, цифры и нижнее подчёркивание

Зададим наш терминальный алфавит $\Sigma = \Sigma_0 \cup \{ \ , \dots \ , \ }$, то есть это Σ_0 и все пробельные символы

Нетерминальный алфавит $N = \{s_0, E_0, V_0, D_0, _ \}$, где s_0 - стартовое состояние, E_0 - это Expr, V_0 - это Var и Ident (в данном случае это одно и то же), D_0 - это числа, $_$ - все пробельные символы (причём давайте считать, что это всевозможные их комбинации, то есть может быть несколько пробелов подряд)

!Важное замечание про пробелы! : все токены разделены пробелами, о есть ключевые слова, переменные, операторы и числа. Далее в грамматике указаны пробелы только внутри одной конструкции, но между конструкциями пробелы тоже обязаны быть.

В нашем языке есть ключевые слова **If, While, Assign, Read, Write и Seq, Nop** (Nop - пустая инструкция)

Для каждого из них введём правило:

- $s_0 \rightarrow If_E_0_s_0_s_0$
(Порядок веток прямой, соответственно если условие E_0 выполнится, то будет исполняться первый s_0 , иначе второй)
- $s_0 \rightarrow While_E_0_s_0$
- $s_0 \rightarrow Assign_V_0_E_0$
- $s_0 \rightarrow V_0_Read$
- $s_0 \rightarrow Write_E_0$
- $s_0 \rightarrow Seq_s_0_s_0$
(Порядок прямой, соответственно вначале идёт первое действие, затем второе)
- $s_0 \rightarrow Nop$
(пустая инструкция)

Выражения:

Правила для бинарных операторов (порядок везде прямой):

- $E_0 \rightarrow -_E_0_E_0$
(Из первого E_0 вычитаем второе E_0)
- $E_0 \rightarrow +_E_0_E_0$
...
- $E_0 \rightarrow \&\&_E_0_E_0$

- $E_0 \rightarrow || _ E_0 _ E_0$

Правила для унарных операторов (унарный минус и отрицание):

- $E_0 \rightarrow -- _ E_0$ (унарный минус)
- $E_0 \rightarrow ! _ E_0$ (отрицание)

Правила для чисел и переменных:

- $E_0 \rightarrow D_0$
- $E_0 \rightarrow V_0$

Переменные и идентификаторы - это строки, которые начинаются обязательно с буквы либо нижнего подчёркивания и состоят из символов Σ_0 , причём обязательно **НЕ** совпадают с ключевыми словами

- $V_0 \rightarrow (a|b|...|_|_a|...|aa|ab|...|Alya_0|.....|Fkb1aa1c_d3|...) \Sigma_0^*$

(Перечисляем все корректные слова длины < 15 (**кроме ключевых слов**) и дальше добавляем что угодно, то есть Σ_0^*)

Числа:

- $D_0 \rightarrow (0|1|...|9)^+$

Пробелы:

- $_ \rightarrow (\backslash t \mid \backslash n \mid \dots)^+$ (все пробельные символы и любые их комбинации)

UPD: 16.04.2020

Функции

Появились новые ключевые слова: **Fun**, **With**, **ThatsAll**, **Call**, **Return**, **WithFun**, **ThatsAllFun**

Fun - объявление функции

With - конкатенация двух списков аргументов

ThatsAll - синоним With , но только с нулём списков

Call - вызов функции

Return - вернуть значение из функции

WithFun - как Seq только для функций, то есть просто их перечисление, принимает два

списка функций

ThatsAllFun - синоним WithFun , но только с нулём списков

Формально:

- $Arg \rightarrow ThatsAll \mid With_V_0_Arg$
(список аргументов)
Примеры:
ThatsAll
With kek With mem ThatsAll
With kek ThatsAll
(то есть список всегда заканчивается ThatsAll-ом)
- $F_0 \rightarrow Fun_V_0_Arg_s_0$
(Объявление функции, это НЕ часть LAst, поэтому нельзя например объявлять функцию внутри функции)
- $ArgCall \rightarrow ThatsAll \mid With_E_0_Arg$
(список аргументов, но уже для вызова функции, то есть вместо переменных тут выражения)
Примеры:
ThatsAll
With 1 With 10 ThatsAll
With + 1 2 ThatsAll
(то есть список всегда заканчивается ThatsAll-ом)
- $E_0 \rightarrow Call_V_0_ArgCall$
(Вызов функции, это часть Ast)
- $s_0 \rightarrow Return_E_0$
(return, это часть LAst)
- $FunList \rightarrow ThatsAllFun \mid WithFun_F_0_FunList$
(Это просто список функций, всегда оканчивается на ThatsAllFun)

Корректная программа на языке L

Корректным кодом на нашем языке является выражение вида:

$Start \rightarrow FunList_s_0$