Формальные языки

Спецификация языка L

Дмитрий Орехов

1 Литералы

Literal

- NumericLiteral

NumericLiteral: Знак минус (опционально), последовательность из хотя бы одной цифры:
-?(\d)+

2 Ключевые слова

Ключевые слова языка.
Keyword

- if
- then
- else
- while
- read
- write
- def
- return

3 Идентификаторы

Идентификатор является последовательностью букв латинского алфавита, нижних подчеркиваний и цифр, причем начинаться должен с буквы или нижнего подчеркивания. Идентификатор также не может являться ключевым словом языка.

Ident:
IdentSequence but not a Keyword

IdentSequence:
[a-zA-Z_]+[a-zA-Z_\d]*

4 Операторы выражений

Операторы выражений, делятся на три группы: булевы операторы, операторы сравнения, арифметические операторы. Приведены с приоритетом и ассоциативностью.

Operator:

- BooleanOperator
- ComparisonOperator
- ComputeOperator

BooleanOperator:

- && (2 RightAssoc)
- || (1 RightAssoc)

ComparisonOperator:

- ==, /=, <=, <, >=, > (3 NoAssoc)

ComputeOperator:

- ^(6 RightAssoc)
- *, / (5 LeftAssoc)
- +, - (4 LeftAssoc)

5 Выражения

Выражения языка также делятся на три типа.
Перед и после выражений может идти произвольное число пробелов.
Expression:

- BooleanExpression
- ComparisonExpression
- ComputeExpression

Идентификаторы являются булевыми выражениями, об их evaluation в комментарии 1. Цепочки булевых выражений или выражений сравнения, соединенных булевыми операторами, являются булевыми выражениями. Булевы выражения могут быть обрамлены круглыми скобками, они влияют на приоритет булевых операторов.

BooleanExpression:

- Ident
- (BooleanExpression)
- BooleanExpression BooleanOperator BooleanExpression
- BooleanExpression *or* ComparisonExpression BooleanOperator BooleanExpression *or* ComparisonExpression
- ComputeExpression

Идентификаторы и выражения сравнения раскрываются в два арифметических выражений, между которыми мы проводим некоторую операцию сравнения. Выражения сравнения могут быть обрамлены круглыми скобками. ComparisonExpresion:

- (ComparisonExpression)
- ComputeExpression ComparisonOperator ComputeExpression

Идентификаторы и числа являются арифметическими выражениями. Арифметические выражения могут быть обрамлены круглыми скобками, обозначающими приоритет операций. Арифмитические выражений рекурсивно раскрываются в два арифмитических выражения, над которыми проводится бинарная операция арифметическим оператором. ComputeExpression:

- NumericLiteral
- Ident
- FunctionCall
- (ComputeExpression)
- ComputeExpression ComputeOperator ComputeExpression

О пробельных символах и операторах выражений:
Вокруг операторов ставится произвольное число пробельных символов: пробелов, табов и переносов строки.
О пробельных символах внутри выражений:
Внутри выражений, между скобками, тоже может быть произвольное число пробельных символов, то есть "(1 + 20) это корректное выражении.
Однако "1 + 2 нет. Я это сделал так, потому что в языке само выражение, в любом случае, не может являться отдельным statement. Поэтому нет смысла, на верхний уровень парсера выражений вытаскивать парсер пробелов.
На уровне языка выражения вида "x = 1 + 2"запарсятся корректно, благодаря уже парсеру statementов, но не expressionов.

Комментарий 1:
Ident встречается у меня как в BooleanExpression, так и в ComputeExpression. Оценивать значения идентификаторов в BooleanExpression я буду как в Python:

- Если идентификатор, хранящий натуральное число, стоит в BooleanExpression, то это False, если там 0, в остальных случаях True.

6 Операторы statements

Операторы, использующиеся в statements языка. Пока что только оператор присваивания.
StatementOperator:

- =

7 Statements[1]

Statement являются: присвоения значения в переменную, условное выражение, while-цикл, операции чтения и записи, последовательность Statements.
Перед и после Statement может быть произвольное число пробелов или табов, разделять разные statement можно символами \n.

Statement:

- Assignment
- IfThenElse
- WhileLoop
- Read
- Write
- Return
- Statement Statement
- Empty - путая строка или произвольная последовательность пробельных символов.

О пробельных символах внутри Statement:
Внутри Statement, в целях удобства форматирования кода, можно ставить произвольное число пробельных символов (пробелы, табы, переносы строки).

8 Присвивание

Имя переменной связывается с численным значением с помощью оператора =. После инструкции ставится ;. Число пробелов вокруг оператора присваивания произвольное, прочие пробельные символы недопустимы.
Assignment:
Ident = ComputeExpression;

[1]Не придумал адекватного перевода на русский язык

9 Условные выражения

После if стоит Expression в круглых скобках, блоки Statement обрамлены фигурными скобками.
Число разделяющих пробелов внутри конструкции произвольное.
IfThenElse:
if (Expression) then {Statement} else {Statement}
if (Expression) then {Statement} else {Statement}

10 While цил

После if стоит Expression в круглых скобках, Statement блок обрамлен фигурными скобками.
Число разделяющих пробелов внутри конструкции произвольное.
WhileLoop:
while (Expression) {Statement}

11 Чтение

Операция чтения значения из потока ввода в идентификатор. Идентификатор обрамляется круглыми скобками. Между read и (Ident) от одного пробела. В конце ставится ;.
Read:
read (Ident);

12 Запись

Операция записи результата выражения в поток вывода. Выражение обрамляется круглыми скобками. Между write и (Expression) от одного пробела.
Write:
write (Expression);

13 Возврат значения

Возвращает результат вычисления выражения из функции.

Return:
return (Expression);

14 Определение функций

Определение функции начинается с ключевого слова def, далее следует ее имя, не являющееся ключевым словом, после в круглых скобках список аргументов через запятую, наконец, тело в фигурных скобках. Везде произвольное число пробельных символов.

FuncDef:

def (Ident not main) (ArgList) {Statement}

ArgList:
Ident
Ident, ArgListContinuation
Empty

ArgListContinuation:
Ident
Ident, ArgListContinuation

15 Вызов функий внутри выражений

Вызов функции не является statement.
FunctionCall:
Ident(ArgList)

16 Программа

Программа - это набор функций и точка входа, функция main без аргументов. Одноименные функции запрещены.

Program:
BeforeMain

BeforeMain:
FuncDef BeforeMain
Main AfterMain

AfterMain:
FuncDef AfterMain
Empty

Main:
def main() {Statement}

17 Источники вдохновения

1. Java Grammar
2. Python Grammar