DSL MMADL

Meta Math Algorithm Description Language

Команда: Алексеев, Куксенко

|  |  |
| --- | --- |
| **Алгоритм**: Заголовок Тело  Заголовок: Имя Параметры  Тело: Операторы  **Фрагмент**: Тело  Тело: Операторы  **Выражение**: Тело  Тело: Операторы | Левый отступ при вложении = 2 пробела  \hspace{}  Перенос при превышении количества символов в строке – новая строка + 4 пробела вправо  (шрифт моноширинный)  (алгоритм, фрагмент)  Выражение – Множество операторов, записанных в одну строку |
| Переменные не обязаны иметь определение | Без явного определения локальных и глобальных переменных |
| Пошаговое уточнение + комментарии | Ключевые слова – синие, жирный шрифт, код – чёрный, комментарий – зелёный |
| **Указание транслятору**  Выход  algorithm - алгоритм  fragment - фрагмент  expression – выражение  Целевой платформы  tex  rtf  html  Стиль  double  python  (По умолчанию Algorithm, tex, double) | Два стиля кода:  Парные границы (double)  Как в Python (python)  **Целевая платформа**  tex (algorithmicx?) (просто tex)  rtf  html |

Лексика:

a, …, z, A, …, Z, 0, …, 9

<-

<=, =>, =, !=, (, ), [, ]

+, -, \*, /, %

&&, ||, |

Вход, Выход

\in, \intersection, \union

for, do, end for, if, then, elseif, else, end if, next for, select, while, end while

return, yield, continue, break

false, true

array, of

\pointer

nil,

:, ;

**Пошаговое уточнение**

Программа -> {(Правило подстановки | Комментарий)}

Правило подстановки -> Левая часть := Правая часть

Левая часть -> <Строка>

Правая Часть -> <\begin> {Код | Комментарий | Левая часть} <\end>

Код -> Строка

Комментарий -> $$ Строка $$

Строка -> {a | … | z | A | … | Z | 0 | … | 9 | \_ | |}

**Алгоритм**

Алгоритм -> Заголовок Тело

Заголовок -> Вход Входные параметры Выход Выходные параметры

Вход -> *Вход:*

Выход -> *Выход:*

Входные параметры -> {Имя параметра*:* Тип параметра \* ,}

Выходные параметры -> {Тип параметра \* ,}

Тело -> {Оператор | Комментарий}

Комментарий -> </b> {a | … | z | A | … | Z | 0 | … | 9 | а | … | я | А | … | Я} </e> //ещё?

Оператор -> Простой оператор | Оператор ветвления // (Составной оператор)

Простой оператор -> Присвоение | Оператор управления | Определение переменной | Выражение

Оператор ветвления -> Условный оператор | Оператор цикла

Присвоение -> Переменная [*:* Тип параметра] Знак присвоения (Выражение | Переменная)

Оператор управления -> Оператор выхода | Оператор управления в цикле

Определение переменной -> Переменная : Тип параметра

Оператор выхода -> (*return* | *yield*) ε |{Выходное значение \* *,*}

Выходное значение -> Выражение | Условие

Оператор управления в цикле -> *continue* | *break*

Выражение -> Математическое выражение | Вызов функции

Вызов функции -> Строка *(*Список переменных*)*

Список переменных -> {Переменная \* ,}

Переменная -> Строка

Строка -> (a | … | z | A | ... | Z) {a | … | z | A | … | Z | 0 | … | 9 }

Математическое выражение -> $ Строка $ // Просто выражение

ИЛИ

Математическое выражение -> (Префиксное выражение | Постфиксное выражение | Бинарное выражение | Переменная | Определенное значение) |

( (Префиксное выражение | Постфиксное выражение | Бинарное выражение | Переменная) )

Префиксное выражение -> Префиксный оператор Математическое выражение

Постфиксное выражение -> Математическое выражение Постфиксный оператор

Бинарное выражение -> Математическое выражение Бинарный оператор Математическое выражение

Определённое значение -> Булево значение | nil

Булево значение -> False | True

Префиксный оператор -> -

Постфиксный оператор -> !

Бинарный оператор -> + | - | \* | \ | % | \intersection | \union | Отношение порядка | Отношение включения

Оператор цикла -> while Условие do Тело end while |

for Диапазон do Тело end for

Условие -> {Одиночное условие \* (and | & | or | | | xor) }

Диапазон -> Отношение включение | Итерация

Одиночное условие -> Булево значение | Отношение порядка

Отношение порядка -> Математическое выражение (= | != | < | > | <= | >=) Математическое выражение

Отношение включения -> Переменная \in Математическое выражение

Итерация -> Переменная from Математическое выражение (to | downto) Математическое выражение

Условный оператор -> if Условие then Тело {ε \* (elseif Условие then Тело ) } [else Тело] end if

ПОКА ФОРМУЛЫ НЕ РАЗБИРАЕМ!