**The Tiny Language**

Program  
 : Declaration { Declaration }

Программа представляет собой последовательность объявлений.

Declaration  
 : Function | Variable

В языке определены две основные сущности: переменные и функции.

Function  
 : **fun** identifier **(** [ Parameters ] **)** [ **:** Type ] **do** Body **end**

Функции могут принимать несколько параметров (или не принимать параметров) и могут возвращать результат. Тип результата (если результат ожидается) задается после списка параметров. Алгоритм функции определяется ее телом. Вызов (активация) функции реализуется посредством конструкции Call в выражении либо оператором вызова.

Parameters  
 : identifier **:** Type { **,** identifier **:** Type }

Параметры функции задают имена, типы и количество аргументов, которые можно передать в функцию при ее вызове. Синтаксис параметров функции подобен объявлениям переменных.

Body  
 : { Variable | Statement }

Тело (алгоритм) функции представляет собой последовательность объявлений переменных и/или операторов, возможно пустую.

Variable  
 : ( **const** | **var** ) identifier **:** Type [ **:=** Expression ]

Объявление переменной задает ее имя, тип, начальное значение, а также допустимость изменения ее значений (константность). Если переменная объявлена как константная, то наличие инициализатора считается обязательным.

Statement  
 : Assignment | Call | Conditional | Loop | Return | Print

Оператор задает некоторое действие в программе. Ассортимент операторов покрывает минимально необходимое множество действий, типичных для разработки несложных программ.

Assignment  
 : Name **:=** Expression

Присваивание назначает новое значение переменной, заданной в его левой части. В качестве левой части допускаются либо переменные, либо элементы массивов.

Call  
 : identifier **(** [ Expression { **,** Expression } ] **)**

Оператор вызова активирует выполнение функции, имя которой задано перед круглыми скобками. Аргументы, передаваемые в функцию, задаются выражениями, разделенными запятыми. Если функция объявлена как не принимающая параметров, (пустые) круглые скобки должны быть сохранены в конструкции.

Conditional  
 : **if** Expression **do** Body [ **else** Body ] **end**

Условный оператор задает выполнение одной из двух последовательностей действий в зависимости от истинности или ложности условия, указанного после служебного слова if. Если значение условия ложно, а компонент else не задан, оператор не совершает никаких действий.

Loop  
 : **while** Expression **do** Body **end**

Оператор цикла задает повторяющееся выполнение действий, заданных в его теле. Выполнение повторяется, пока условие, заданное после служебного слова while, истинно. Условие проверяется перед каждой очередной итерацией цикла. Если перед самой первой итерацией значение условия ложно, то оператор не совершает никаких действий.

Print  
 : **print** Expression

Оператор выводит на консоль значение выражения, заданного после служебного слова **print**.

Return  
 : **return** [ Expression ]

Expression  
 : Logical [ ( **<** | **<=** | **>** | **>=** | **=** | **/=** ) Logical ]

Выражение задает формулу для вычисления некоторого значения. В целом, синтаксис выражений определяется по традиционным правилам, с учетом обычных приоритетов операций.

Logical  
 : Factor { ( **and** | **or** | **xor** ) Factor }

Factor  
 : Term { ( **+** | **-** ) Term }

Term  
 : Primary { ( **\*** | **/** ) Primary }

Primary  
 : [ **+** | **-** ] ( literal | Name | Call | **(** Expression **)** )

Name  
 : identifier [ **[** Expression **]** ]

Литералы (константы) целого типа задаются в виде непустой слитной последовательности десятичных цифр. Литералы вещественного типа представляют собой композицию из литерала целого типа (целая часть вещественного), символа точки, за которой следует литерал целого типа, представляющий дробную часть вещественного литерала.

Идентификатор определяется как непустая слитная последовательность букв и цифр, начинающаяся с буквы.

Type  
 : SimpleType [ **[** [ Expression ] **]** ]

SimpleType  
 : **integer** | **real** | **boolean**

В языке определены три базовых типа – целый, вещественный и булевский, которые задаются посредством служебных слов **integer**, **real** и **boolean**, соответственно. Для целых и вещественных значений определен традиционный набор операций – сложение, вычитание, умножение и деление, а также шесть обычных операций сравнения. Булевский тип представлен двумя значениями истинности **true** и **false**. Для значений булевского типа определены логические операции **or**, **and** и **xor**, семантика которых традиционна. Результат этих операций, а также результат операций сравнения по определению имеет булевский тип.