Лексический синтаксис

1. <ident> Идентификатор — непустая последовательность букв латинского алфавита в любом регистре, цифр и символа нижнего подчеркивания (_), начинающаяся на букву латинского алфавита в нижнем регистре, не являющаяся ключевым словом.
   * Корректные идентификаторы: x, list, listNat_123.
   * Некорректные идентификаторы: Abc, 123, _List.

2. func <ident>(<vars>){<body>}Функция непустая последовательность букв латинского алфавита в любом регистре, цифр и символа нижнего подчеркивания (_), начинающаяся на букву латинского алфавита в нижнем регистре, не являющаяся ключевым словом. Содержит в себе список переменных <vars> и тело функции <body>.

3. <vars> — список, состоящий из неотрицательной последовательности идентификаторов, разделенных запятой <,>.

4. <body> — список, состоящий из неотрицательной последовательности инструкций, разделенных точкой с запятой <;>.

5. <keyword> слова не могут быть идентификаторами. Конкретные ключевые слова вы выбираете сами.

6. <num> Число: натуральное или ноль в десятичной системе счисления, не может содержать лидирующие нули.
   * Корректные числа: 123, 0.
   * Некорректные числа: -1, 007, 89A.

7. <operator> Операторы языка:
   * минус -, унарная
   * возведение в степень **, бинарная, правоассоциативная
   * умножение *, бинарная, левоассоциативная
   * деление /, бинарная, левоассоциативная
   * сложение +, бинарная, левоассоциативная
   * вычитание -, бинарная, левоассоциативная
   * сравнение <, <=, ==, /=, >, >=, бинарная, неассоциативная
   * логическое отрицание --, унарная
   * конъюнкция &&, бинарная, правоассоциативная
   * дизъюнкция ||, бинарная, правоассоциативная
   * оператор присвоения =, бинарная, правоассоциативная
   * оператор конца инструкции;
8. Пробелы и символы переноса строки не являются значимыми, но не могут встречаться внутри одной лексемы. Строка, содержащая только пробелы, называется пустой строкой, и анализатор полностью игнорирует её.

Конкретный синтаксис

1. Ключевые слова:

   return - инструкция для возвращения значения из функции;

   if - инструкция для определения ветки с условным выражением;

   else - инструкция для определения опциональной ветки условного выражения ***while*** - инструкция для определения цикла с предусловием;

   repeat - инструкция для определения цикла с постусловием;

   id - инструкция для инициализации переменной(<ident>);

func - инструкция для инициализации функции;

2. Определение функции содержит ее сигнатуру и тело. Сигнатура функции содержит ее название (идентификатор) и список аргументов (может быть пустым). Тело — последовательность инструкций (может быть пустой).

```
func <ident>(<ident>, <ident>, ...){
<body>}
```

Тело функции указывается в фигурных скобках {}.

Примеры:

```
func sum(a, b){
    return a + b;
}

func fib(n){
    if(n == 1 || n == 2){
        return 1;}
    if(n == 0){
        return 0;}
    id f_1 = 0;
    id f_2 = 1;
    id i = 1;
    while(i <= n){
        f_1 = f_2 + f_1;
        f_2 = f_1 – f_2;
        i = i + 1;
        return f_1;}
    }
```

3. Программа — непустая последовательность определений функций, инициализаций переменных и их взаимодействий с помощью операторов и инструкций.

```
func <ident>(<ident>, <ident>, ...){
<body>}
```

id <ident>=<ident>(<num>);

…

4. Инструкции:
    - Присвоение значения арифметического выражения переменной. Переменная может быть произвольным идентификатором.

        Примеры:

        a = 1 + 2;

        a = b * c;
    - Возвращение значения из функции.

        Примеры:

        return a;

        return 1 + 2;
    - Условное выражение с обязательной веткой else. Условием является арифметическое выражение в круглых скобках (). Условие не может быть пустым. Тело условного выражения — произвольная последовательность инструкций (может быть пустой) и указывается в фигурных скобках {}.

        Пример:

        if(n == 1 || n == 2){return 1;}

        else{ return 2;}

    - Цикл с предусловием while. Условием является арифметическое выражение в круглых скобках (). Тело цикла — произвольная последовательность инструкций (может быть пустой) и указывается в фигурных скобках {}.

Пример:

```
id i = 0;
while(i <= 5){
f_1 = f_2 + f_1;
f_2 = f_1 – f_2;
i = i + 1;}
```

- Цикл с постусловием repeat. Условием является арифметическое выражение в круглых скобках (). Тело цикла — произвольная последовательность инструкций (может быть пустой) и указывается в фигурных скобках {}.

  Пример:

```
id i = 0;
repeat(i <= n){
f_1 = f_2 + f_1;
f_2 = f_1 – f_2;
i = i + 1;}
```