Документация Daiquiri Language

Ключевые слова

Ключевые слова можно писать с заглавной или строчной буквы.

Слово	**Применение**
Вывод (вывод)	Функция для вывода
Если (если)	Конструкция условие
Иначе (иначе)	Конструкция при невыполнении условия
Пока (пока)	Конструкция цикла с предусловием
Для (для)	Конструкция цикла с инициализацией, условием и обновлением
Делать (делать)	Конструкция цикла «Делать-Пока» с постусловием
Остановить (остановить)	Конструкция для завершения итерации цикла
Продолжить (продолжить)	Конструкция для остановки цикла
не	Логическое не (отрицание)

Операторы

Числовые бинарные	
+	сложение
-	вычитание
*	умножение
/	деление
Унарные	
+	унарный плюс
-	унарный минус
Логические бинарные	
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно
==	равно
!=	не равно
&&	и
||	или

Другие	
//	комментарий
()	скобки
[]	элемент по индексу
{}	блоки
=	оператор присвоения

Переменные

Синтаксис.

```
имя переменной = [значение]
```

Примеры:

```
переменная1 = "Daiquiri – это язык программирования"

переменная2 = 132
```

Переменные могут быть либо числом (по умолчанию double) или

Константы, встроенные в язык.

Имя константы	Значение
ПИ (PI)	Число пи 3.14…
E (E лат)	Число e 2.71…
истина	1
ложь	0

Литералы.

Литерал	Назначение
\н (\n)	Перенос строки

\т (\t)	Табуляции
\	Экранирование символа, например \" или \\

Функция вывода

Синтаксис.

```
Вывод [выражени, строка, число, константа или переменная]
```

В выводе доступна конкатенация строк.

Пример:

```
переменная = 1001

Вывод "Значение переменной = " + переменная + "\н"
```

Ветвления

Синтаксис.

```
Если [условие] {

    [код]

} Иначе {

    [код]

}
```

Пример.

```
Флаг = 12
Если Флаг > 0 {
    Флаг = Флаг + 1
    Вывод Флаг + "\н"
} Иначе {
    Флаг = 0
    Вывод Флаг + "\н"
}
```

Исполнение кода при верном и ложном условии.

Когда блок [код] содержит только одну строку, то можно упустить фигурные скобки «{}».

Строковый тип

Достать элемент по индексу.

Чтобы достать элемент строки по индексу нужно использовать оператор «[]». После строки или переменной в квадратных скобках указывается индекс символа, который нужно получить, отчет индексов в строке начинаются с 0.

Пример получения символа "а" из строки:

```
переменная = "строка"
Вывод переменная[5] + "\н"
// или
Вывод "строка"[5] + "\н"
```

Получение подстроки.

Также с помощью оператора «[]» доступно получение подстроки, для этого нужно указать в скобках индекс начало подстроки, которую нужно получить, а затем через символ двоеточия «:» указать индекс конца подстроки.

[индекс начала : индекс конца]

Можно упускать индекс начала или индекс конца подстроки, тогда за индексы начала или конца будет браться начало и конец исходной строки. Также можно упустить оба индекса, но поставив двоеточие с квадратных скобках, тогда вернется исходная строка.

Примеры получения подстроки:

```
переменная = "строка"
Вывод переменная[2 : 4] + "\н"
// будет выведено "ро"
Вывод переменная [2 : ] + "\н"
// будет выведено "рока"
Вывод переменная [ : 3] + "\н"
// будет выведено "стр"
Вывод переменная [ : ] + "\н"
// будет выведено "строка"
```

Циклы

Цикл «Пока» («пока») цикл с предусловием.

Блок цикла [код] выполняется пока, верно [условие].

Синтаксис.

```
Пока [условие] {
    [код]
}
```

Пример.

```
и = 0
Пока и < 10 {
    Вывод и + "\н"
    и = и + 1
}
```

Цикл «Делать-Пока» («делать - пока»).

Блок цикла [код] выполняется пока, верно [условие].

Синтаксис.

```
Делать {
    [код]
} Пока [условие]
```

Пример.

```
и = 0
Делать{
        Вывод и
        Вывод "\н"
        и = и + 1
} Пока (и < 10)
```

Цикл «Для» («для»).

Цикл состоит из блоков инициализации, условие и обновление:

- Инициализация: выполняется один раз при начале цикла и используется для инициализации переменной цикла.
- Условие: определяет условие, при котором цикл будет продолжаться. Если условие истинно, выполнение кода в цикле продолжается, иначе цикл завершается.
- Обновление: выполняется после каждой итерации цикла и обычно используется для изменения переменной цикла.

Если блок [код] содержит одну строку фигурные «{}» скобки можно упустить.

Синтаксис.

```
Для ([инициализация]; [условие]; [обновление]) {
        [код]
}
```

Пример.

```
Для (и = 0; и < 10; и = и + 1)
    Вывод "итерация: " + и + "\н"
```

Остановить и Продолжить (остановить, продолжить).

Оператор «Остановить» используется для прерывания выполнения цикла. Когда оператор «Остановить» встречается внутри цикла, выполнение цикла прекращается, и управление передается к следующему оператору после цикла.

Оператор «Продолжить» используется для пропуска текущей итерации цикла и перейти к следующей итерации. Когда оператор «Продолжить» встречается внутри цикла, код ниже оператора «Продолжить» в текущей итерации не будет выполнен, и управление передастся следующей итерации цикла.

Функции

Системные функции.

Функция	**Назначение**
длина([строка])	Получение значения длины строки
индекс([строка], [символ или подстрока], [начальный индекс])	Используется для поиска индекса первого вхождения указанного символа или подстроки в строке. Этот метод возвращает индекс первого вхождения указанного символа или подстроки в строке, или -1, если символ или подстрока не были найдены. Также можно указать третьем параметром начальный индекс, с которого будет начинаться поиск.

синус([число])	Получение значения синуса от числа
косинус([число])	Получение значения косинуса от числа