

Дожди́ков И́горь

Руководство пользователя

KolibriMath

Версия 1.2.1

Киров 2021

В данном руководстве представлены основные моменты использования языка вычислений KolibriMath. Каждый раздел сопровождается простыми примерами и иллюстрациями, позволяющими лучше усвоить подаваемый материал.

(С) Дождиков И. С., 2021.

Содержание

Введение	4
Внутреннее устройство языка KolibriMath	5
Основы языка KolibriMath	6
Основные типы данных	6
Основные конструкции.....	8
Встроенные функции	9
Вывод.....	10
Скрипты.....	11
Обработка ошибок	11
Заключение.....	14
Ссылки	15

Введение

Идея разработать язык вычислений пришла мне во время разработки первого примера для операционной системы Kolibri. Сперва это задумывалось как простой язык вычислений, но я не думал, что он вырастет в нечто большее, чем просто «сложитель» чисел. Вообще говоря, это мой первый проект реализации текстового языка программирования с нуля и до такого масштаба. Надеюсь, он заинтересует Вас, и Вы тоже внесете вклад в сообщество Kolibri. Спасибо всем, кто помогал мне с этим проектом, без Вашей помощи я бы добирался до итогов намного дольше, чем есть сейчас.

1 Внутреннее устройство языка KolibriMath

Перед тем, как глубже копнуть в сторону KolibriMath, следует задаться вопросом, что же такое KolibriMath?

KolibriMath – интерпретируемый язык вычислений с динамической типизацией. Слово «интерпретируемый» говорит о том, что он выполняется непосредственно по ходу чтения программой-интерпретатором исходного кода на языке KolibriMath. Словосочетание «динамическая типизация» говорит о том, что в языке переменные могут менять свой тип по ходу вычислений.

Так как KolibriMath – язык интерпретируемый, то из архитектуры проекта сразу был исключен генератор кода. Тогда архитектуру самого интерпретатора можно представить следующим образом (рисунок 1).

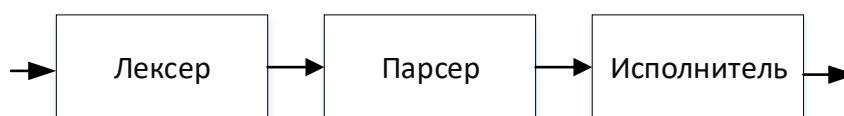


Рисунок 1 – Архитектура интерпретатора

Из рисунка 1 видно, что всего у интерпретатора имеет три модуля.

На вход лексера поступает исходный текст программы на языке KolibriMath, причем это может быть строка из консоли, так и текстовый файл. Лексер производит лексический анализ и выделяет в тексте программы токены. Список токенов передается на вход парсера.

Парсер представляет собой модуль, который строит из строки токенов абстрактное синтаксическое дерево программы. С выхода парсера абстрактное синтаксическое дерево подается на вход исполнителя.

Исполнитель осуществляет исполнение абстрактного синтаксического дерева и выдает результат пользователю. Исполнитель содержит таблицу переменных, используемых программой.

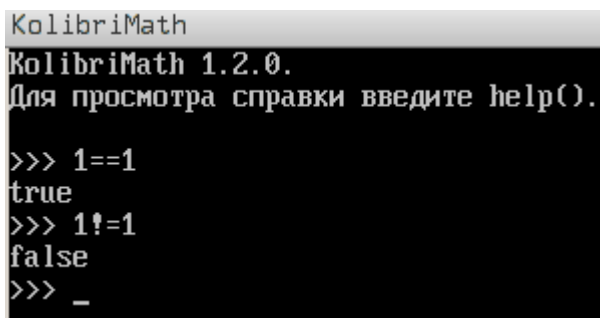
2 Основы языка KolibriMath

2.1 Основные типы данных

Язык KolibriMath поддерживает шесть типов данных: целое число, дробное число, логическое значение, целочисленная матрица и дробная матрица, строковая константа.

С целыми и дробными числами можно производить операции сложения (“+”), вычитания (“-“), умножения (“*”) и деления (“/”).

Логический тип представляет собой тип, который получается неявным преобразованием из целого типа. То есть число “1” соответствует значению “true” логического типа, а число “0” – значению “false” логического типа. Причем, при использовании логического выражения в качестве входа, на выходе мы получаем результат в виде логического выражения, а не числа (рисунок 2).



```
KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> 1==1
true
>>> 1!=1
false
>>> _
```

Рисунок 2 – Результат логического выражения

Логическими операциями в рамках KolibriMath являются: сравнение на равенство (“==”) и неравенство (“!=”), сравнение на величину (“<”, “>”, “>=”, “<=”), доступна операция инвертирования (“!”). По поводу операции инвертирования можно сказать, что она выдает в качестве результата не логическое значение, а обычное число.

Целочисленная и дробная матрицы представляют собой двумерные массивы содержащие соответствующие значения своего типа. Значение типа матрицы жестко фиксировано, и если пользователь попытается, например, присвоить значение целого типа элементу дробной матрицы, то он получит ошибку (рисунок 3).

```
KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> floatMatrix('a',1,1)

>>> a[0,0]=1

произошла ошибка: изменение невозможно, так как переменная принадлежит другому т
ипу [код: 5.6].

>>> _
```

Рисунок 3 – Результат присваивания дробной матрице целочисленного элемента

Матрицы инициализируются значениями так, как показано на рисунке 3, то есть через специальные функции “intMatrix” и “floatMatrix”. Размер матриц можно изменить с помощью функции “resizeMatrix”, которая принимает в качестве аргументов название переменной, новое число строк и новое число столбцов. Пример изменения размера матрицы представлен на рисунке 4.

```
KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> intMatrix('a',1,1)

>>> a
0

>>> resizeMatrix('a',3,3)

>>> a
0      0      0
0      0      0
0      0      0

>>> _
```

Рисунок 4 – Пример изменения размера матрицы

Матрицы можно складывать (“+”) и вычитать (“-“) между собой, умножать на число (“*”), умножать друг на друга (“**”). Пример работы с матрицами представлен на рисунке 5.

```

KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> intMatrix('a',3,3)

>>> a[0,0]=5

>>> a*a+a
  30    0    0
   0    0    0
   0    0    0

>>> _

```

Рисунок 5 – Пример работы с матрицами

Последним типом языка KolibriMath являются строковые константы. Строковые константы – это строки, заключенные в одинарные кавычки (“ ’ ”). Следует заметить, что они могут использоваться только как самостоятельные значения или как аргументы функций. Переменным нельзя присваивать строковые константы. Пример работы со строковой константой представлен на рисунке 6.

```

KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> 'hello, world!'
hello, world!

>>> _

```

Рисунок 6 – Пример работы со строковой константой

2.2 Основные конструкции

Основными конструкциями языка KolibriMath являются:

- присваивание переменной;
- удаление переменной
- условные конструкции if...elif...else;
- цикл repeat.

Присваивание значения переменной производится через стандартную операцию присваивания (“=”). Можно присвоить переменной целое или дробное значение, целочисленную или дробную матрицу.

Удаление переменной производится через операцию удаления (“del”) (рисунок 7).


```
KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> a=5

>>> del a

>>> a

произошла ошибка: переменная с таким именем не найдена [код: 5.8].

>>> _
```

Рисунок 7 – Удаление переменной

Условие условной конструкции или цикла заключается в круглые скобки. Тело заключается в фигурные скобки. Пример цикла представлен на рисунке 8.

```
KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> a=1

>>> repeat(a<3){a,a=a+1}
12
>>> _
```

Рисунок 8 – Пример цикла

2.3 Встроенные функции

Ранее не было сказано, но в KolibriMath поддерживаются только встроенные функции, пользовательских функций пока нет. Синтаксис вызова функций прост: пишется имя функции, далее в круглых скобках через запятую идут аргументы функции. Аргументами функции могут быть строковые константы, целые и дробные числа, целочисленные и дробные матрицы.

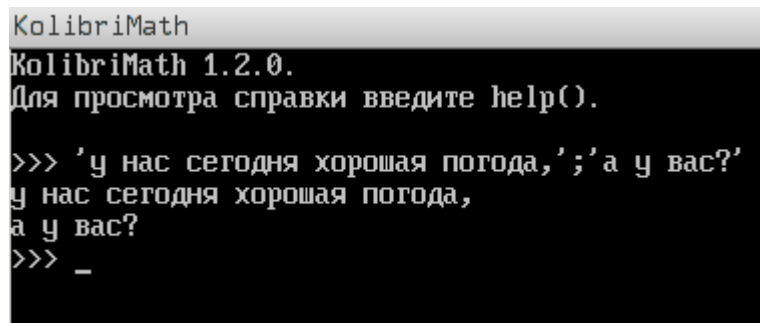
Список встроенных функций представлен ниже (таблица 2).

Таблица 1 - Список встроенных функций

Название функции	Аргументы функции	Описание функции
int	<число>	преобразует число в целое число
float	<число>	преобразует число в дробное число
sqrt	<число>	извлекает корень из числа
pow2	<число>	возводит число в квадрат
pow	<число>, <степень>	возводит число в указанную степень
sin	<число>	находит синус числа
cos	<число>	находит косинус числа
abs	<число>	находит модуль числа
file	<путь>	открывает скрипт для исполнения
input	-	ввод числа в переменную
input	<подсказка>	ввод числа в переменную с подсказкой
ceil	<число>	округляет число в большую сторону
mod	<число>	получает дробную часть числа
intMatrix	<название переменной>, <число строк>, <число столбцов>	инициализирует целочисленную матрицу
floatMatrix	<название переменной>, <число строк>, <число столбцов>	инициализирует дробную матрицу
resizeMatrix	<название переменной>, <новое число строк>, <новое число столбцов>	изменяет размер матрицы
det	<название переменной>	находит определитель матрицы
T	<название переменной>	транспонирует матрицу
inverse	<название переменной>	находит обратную матрицу

2.4 Вывод

Вывод по умолчанию входит в стандартный список возможностей языка KolibriMath. Для вывода достаточно написать имя переменной или строковую константу. Причем важно помнить, что для разделения конструкций используется знаки (“,”) и (“;”). Различие между ними то, что второй знак, помимо разделения инструкций, переводит каретку на новую строку (рисунок 9).



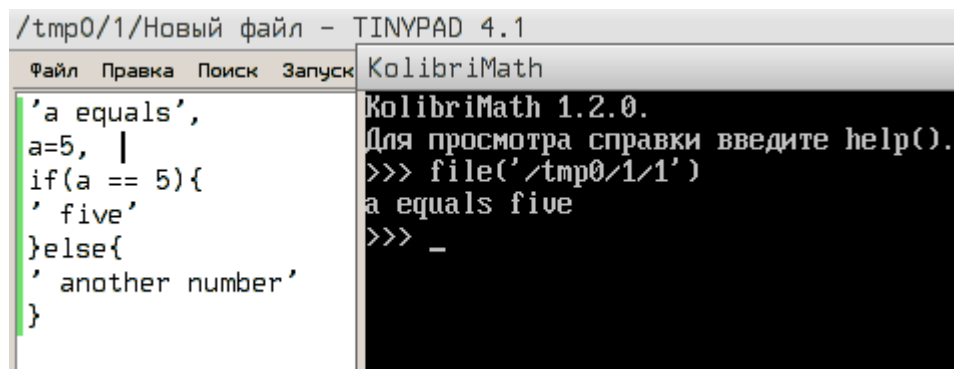
```
KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().

>>> 'у нас сегодня хорошая погода, ';'а у вас?'
у нас сегодня хорошая погода,
а у вас?
>>> _
```

Рисунок 9 – Пример использования знака “;”

2.5 Скрипты

В качестве инструкций могут быть применены не только отдельные инструкции, но и совокупность инструкций в виде файла. Для открытия этого файла используется функция “file” (рисунок 10).



```
/tmp0/1/Новый файл - TINYPAD 4.1
Файл Правка Поиск Запуск KolibriMath
KolibriMath 1.2.0.
Для просмотра справки введите help().
>>> file('/tmp0/1/1')
a equals five
>>> _
```

```
'a equals',
a=5, |
if(a == 5){
' five'
}else{
' another number'
}
```

Рисунок 10 – Пример использования скриптов

2.6 Обработка ошибок

Все ошибки в KolibriMath классифицируются в системе, состоящей из двух групп чисел. Первое число обозначает место возникновения ошибки: 0 – при вводе в консоль, 1 – при работе модуля загрузки файла, 2 – при работе модуля лексера, 3 – при работе модуля парсера, 4 – при работе модуля вывода ошибок, 5 – при работе модуля исполнителя. Второе число обозначает саму ошибку.

Таблица возможных ошибок представлена ниже (таблица 2).

Таблица 1 – Коды ошибок и их описание

Код группы ошибок	Код ошибки	Текстовое описание
0	0	Строка не может быть слишком длинной.
	1	Ошибка выделения памяти.
1	0	Путь не найден.
	1	Невозможно открыть файл.
	2	Файл не может быть слишком большой.
	3	Ошибка выделения памяти.
2	0	Название переменной и функции не должно начинаться с цифры.
	1	Неожиданный символ.
	2	Ошибка выделения памяти.
	3	Ожидалась цифра или '.'.
3	0	При парсинге произошла ошибка.
	1	Неожиданный конец выражения.
	2	Ожидалась операция удаления.
	3	Ожидалось название переменной.
	4	Ожидался символ ',' или конец выражения.
	5	Ожидался символ '='.
	6	Ошибка при получении числа.
	7	Ожидалось название функции.
	8	Неожиданный символ.
	9	Ожидалось число.
	10	Ошибка выделения памяти.
	11	Строка не может быть частью арифметического выражения.
	12	Индекс элемента массив должен быть целым числом.
4	0	Ошибка выделения памяти.
5	0	Ошибка выделения памяти.
	1	Строка не может быть слишком длинной.
	2	Функция с таким числом аргументов не найдена.
	3	Не удалось выполнить функцию.
	4	Функция 'sqrt' принимает отрицательный аргумент.
	5	Функция 'row' принимает нулевой показатель.
	6	Изменение невозможно, так как переменная принадлежит другому типу.
	7	Индекс имеет недопустимый формат.
	8	Переменная с таким именем не найдена.

Продолжение таблицы 2.

5	9	Деление на ноль запрещено.
	10	Результат операции над матрицами не получен.
	11	Данная операция неприменима к матрицам.
	12	Переменная с таким именем уже существует.

Заключение

В этом небольшом руководстве были рассмотрены внутреннее устройство языка KolibriMath, основные типы данных языка KolibriMath, основные конструкции языка KolibriMath, список встроенных функций, особенности вывода и формирования скриптов, список ошибок, выводимых при неправильной работе программы.

Надеюсь, это руководство поможет Вам освоить программирование вычислений на языке KolibriMath.

Ссылки

- | | | | | |
|---|---|---|------|----------|
| 1 | Ссылка | на | тему | проекта: |
| | http://board.kolibrios.org/viewtopic.php?f=9&t=4392 ; | | | |
| 2 | Ссылка на GitHub проекта: | https://github.com/Igoru99/KolibriMath ; | | |
| 3 | Связь с автором: | igoru99@gmail.com . | | |