федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники



Вариант Groovy Fantom Julia

Лабораторная работа №5

по дисциплине

“**информатика**”

Выполнил

Студент группы P3113

Чжоу Хунсян: 336184

Преподаватель:

**Балакшин Павел Валерьевич**

**Table of Contents**

[1. Groovy 3](#_Toc90129548)

[1.1 Общая характеристика и история создания 3](#_Toc90129549)

[1.2. Система типов и особенности архитектуры языка 4](#_Toc90129550)

[1.3. Примеры кода и интересные факты 5](#_Toc90129551)

[2. Fantom 7](#_Toc90129552)

[2.1 Общая характеристика и история создания 7](#_Toc90129553)

[2.2. Система типов и особенности архитектуры языка 7](#_Toc90129554)

[2.3. Примеры кода и интересные факты 7](#_Toc90129555)

[3. Julia 9](#_Toc90129556)

[3.1 Общая характеристика и история создания 9](#_Toc90129557)

[3.2. Система типов и особенности архитектуры языка 9](#_Toc90129558)

[3.3. Примеры кода и интересные факты 11](#_Toc90129559)

[Список использованной литературы 12](#_Toc90129560)

[Опрос 12](#_Toc90129561)

# 1. Groovy

[[1]](#_1._Groovy)

Рис 1‑1

1.1 Общая характеристика и история создания

**Groovy** — объектно-ориентированный язык программирования, разработанный для платформы Java как дополнение к языку Java с возможностями Python, Ruby и Smalltalk.

Использует Java-подобный синтаксис с динамической компиляцией в JVM байт-код и напрямую работает с другим Java-кодом и библиотеками. Язык может использоваться в любом Java-проекте или как скриптовый язык. Активно используется как Предметно-ориентированный язык для написания скриптов при работе в предметной области (от математики до обработки научных данных), а также сборке и тестированию приложений(Gradle).

Groovy завершил процесс стандартизации в Java Community Process (JSR 241).

Основные возможности языка, отличающие его от Java: как статическая, так и динамическая типизация; встроенный синтаксис для списков, ассоциативных массивов, массивов и регулярных выражений; перегрузка операций. Замыкания в Groovy появились задолго до Java.

Первым упоминанием о языке было сообщение в блоге Джеймса Строна (англ. *James Strachan*)[[2]](https://ru.wikipedia.org/wiki/Groovy#cite_note-2) от августа 2003 года. Позднее было выпущено несколько версий между 2004 и 2006 годами. После того, как начался процесс стандартизации JCP, нумерация версий была изменена, и версию называют «1.0». Версия «1.0» была выпущена 2 января 2007 года. В декабре 2007 года вышел Groovy 1.1, эта версия вскоре была перенумерована как «1.5» вследствие значительных изменений в языке.

Стрэчен покинул проект за год до релиза Groovy 1.0 в 2007 году, а в июле 2009 года Стрэчен написал в своём блоге, что возможно не создал бы Groovy, если бы в 2003 году прочитал книгу Мартина Одерского с соавторами о программировании на языке Scala (вышедшую в 2007 году).

Проект разработки языка и комитет JSR-241 с 2007 года возглавляет Гийом Лафорж (*Guillaume Laforge*). Компанию G2One, занимавшуюся развитием и коммерциализацией языка и фреймворка Grails, осенью 2008 года приобрела SpringSource, в итоге через цепочку слияний и поглощений (VMware, Pivotal, EMC) актив перешёл в 2017 году в корпорацию Dell. С 2015 года язык является проектом верхнего уровня фонда Apache.

В 2020 году вышла версия 3.0.0 с целым пакетом улучшений и достоинств:

1. do-while циклы, лямбда выражения, try-with-resources,code blocks (анонимные блоки кода).
2. инициализация массива в "Java style" (new int[] {1, 2, 3}).
3. дефолтные(default) методы в интерфейсе.
4. новые операторы сравнения и Элвис-оператор.
5. инициализация не статических внутренних классов (outer.new Inner())
6. безопасные индексы (nullableVar?[1, 2])

[[1]](#_1._Groovy)

1.2. Система типов и особенности архитектуры языка

**Система типов: статическая и динамическая**

В отличие от Java, в Groovy исходный код может быть выполнен как обычный скрипт, если содержит код вне определения класса или класс с методом main или Runnable или GroovyTestCase:

#!/usr/bin/env groovy

println "I can execute this script now!"

Строки в Groovy: Java Strings с одинарными кавычками и GStrings с двойными кавычками.

def javaStyleString = 'java String style'

def GStringsStyleString = "${javaStyleString}"

def j = '${javaStyleString}'

def bigGroovyString = """

${javaStyleString}

${GStringsStyleString}

"""

println bigGroovyString

[[1]](#_1._Groovy)

1.3. Примеры кода и интересные факты

Классический пример вывода на печать строки с приветствием:

*// Comment*

print("Hello world!")

Пример создания и использования функции с аргументами и возвращаемым значением:

def helloFunction(name) {

println("Hello, ${name}")

}

helloFunction("Groovy!")

*// В следующей функции вычисления суммы двух объектов*

*// последнее выражение является возвращаемым значением*

def sum(a, b) {

a + b *// вернет результат сложения*

}

print(sum(1, 2))

*// Эта функция принимает параметры только числового типа*

def subtraction(double a, double b) {

**return** a - b

}

print(subtraction(6, 4))

Объявление класса, создание его экземпляра и вызов его методов в главной точке входа программы (статический метод main())

**class** **Foo** {

def doSomething() {

def data = ["name": "James", "location": "London"]

**for** (e **in** data) {

println("entry ${e.key} is ${e.value}")

}

}

def closureExample(collection) {

collection.each { println("value ${it}") }

}

**static** void main(args) {

def values = [1, 2, 3, "abc"]

def foo = **new** Foo()

foo.closureExample(values)

foo.doSomething()

}

}

";" — точка с запятой в конце строки не обязательна.

[[1]](#_1._Groovy)

# 2. Fantom

[[2]](#_2._Fantom)

2.1 Общая характеристика и история создания

**FANTOM** (англ. *FANTOM* — Functional Annotation of the Mammalian Genome, рус. *Функциональная аннотация геномов млекопитающих*) — международный исследовательский консорциум, основанный доктором Хаяшизаки и его коллегами в 2000 году с целью функционального аннотирования полноразмерных кДНК, которые были собраны в ходе проекта *Mouse Encyclopedia* в научном центре RIKEN. С тех пор FANTOM стал самостоятельным и развитым проектом, который затрагивает разные сферы анализа транскриптомов. Цель проекта — прийти от понимания «элементов» — транскриптов к понимания «системы» — транскрипционной регуляторной сети.[[2]](#_2._Fantom)

2.2. Система типов и особенности архитектуры языка

Система типов Fantom проста по дизайну. Все переменные статически типизированы, как и в C# и Java. Fantom отклоняет универсальные типы из-за их сложности, но у него есть набор встроенных универсальных типов: List, Map и Func. Fantom также может воспринимать язык с динамической типизацией посредством динамических вызовов и автоматического перехода на более позднюю версию. Fantom имеет простой в использовании api-интерфейс отражения и возможности метапрограммирования.

Fantom является открытым исходным кодом в рамках Academic Free License 3.0 и доступен для платформ windows и Unix-подобных (включая Mac OS X).[[2]](#_2._Fantom)

2.3. Примеры кода и интересные факты

Вот классическая программа Hello world, написанная на языке Фантом:

*{*

**()** **основная**

статическая пустота

**{** Класс HelloWorld

// Привет от Фантом!

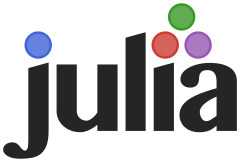
эхо("Привет, Мир!")

}

}

[[2]](#_2._Fantom)

# 3. Julia

[[3]](#_3._Julia)

3.1 Общая характеристика и история создания

Обсуждение необходимости создания нового языка началось в августе 2009 года. Стефан Карпински, к тому времени поработавший с математическим пакетом Matlab и языком программирования R, испытывал неудовлетворённость их ограничениями. И после того, как Вирал Шах познакомил его с Джеффом Безансоном, все трое принялись обсуждать концепцию нового языка. Для выбора женского имени в качестве названия языка, как пояснял в интервью ведущий разработчик языка Стефан Карпински, «особой причины не было», разработчикам просто «понравилось это имя. Первая открытая версия была опубликована в феврале 2012.[[3]](#_3._Julia)

3.2. Система типов и особенности архитектуры языка

|  |  |
| --- | --- |
| Тип данных | Описание |
| Int8 | представляет целые числа со знаком размером не более 8 бит (от -128 до 127) |
| UInt8 | представляет целые положительные числа размером не более 8 бит (от 0 до 255) |
| Int16 | представляет целые числа со знаком размером не более 16 бит (от -32768 до 32767) |
| UInt16 | представляет целые положительные числа размером не более 16 бит (от 0 до 65535) |
| Int32 | представляет целые числа со знаком размером не более 32 бита (от -2147483648 до 2147483647) |
| UInt32 | представляет целые положительные числа размером не более 32 бита (от 0 до 4294967295) |
| Int64 | представляет целые числа со знаком размером не более 64 бита (от -9223372036854775808 до 9223372036854775807) |
| UInt64 | представляет целые положительные числа размером не более 64 бита (от 0 до 18446744073709551615) |
| Int | представляет целые числа со знаком, например, 1, -30, 458. На 32-разрядных платформах эквивалентен Int32, а на 64-разрядных - Int64 |
| UInt | представляет целые положительные числа, например, 1, 30, 458. На 32-разрядных платформах эквивалентен UInt32, а на 64-разрядных - UInt64 |
| Float | 32-битное число с плавающей точкой, содержит до 6 чисел в дробной части |
| Double | 64-битное число с плавающей точкой, содержит до 15 чисел в дробной части |
| Float80 | 80-битное число с плавающей точкой |
| Bool | представляет логическое значение true или false |
| String | представляет строку |
| Character | представляет отдельный символ |

[[3]](#_3._Julia)

* Мультиметод: обеспечивает возможность определять поведение функции в зависимости от типа передаваемых аргументов
* Динамическая типизация
* Хорошая производительность, сравнимая со статически типизированными языками как C
* Встроенная система управления пакетами
* [Макросы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0) и другие возможности метапрограммирования
* Вызов Python функций при помощи PyCall
* Вызов [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) функций напрямую: без дополнительных надстроек и API
* Богатые возможности для управления другими процессами
* Разрабатывался для параллельных и распределенных вычислений
* [Сопрограммы](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0): легковесные зеленые потоки (green threads)
* Возможность определять дополнительные типы не уступающие в скорости и удобстве встроенным
* Элегантные и расширяемые преобразования для числовых и других типов

Поддержка [Юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4), включающая, но не ограничиваемая только UTF-8[[3]](#_3._Julia)

3.3. Примеры кода и интересные факты

Пример функции:

**function** mandel(z)

c = z

maxiter = 80

**for** n = 1:maxiter

**if** abs(z) > 2

**return** n-1

**end**

z = z^2 + c

**end**

**return** maxiter

**end**

**function** randmatstat(t)

n = 5

v = zeros(t)

w = zeros(t)

**for** i = 1:t

a = randn(n,n)

b = randn(n,n)

c = randn(n,n)

d = randn(n,n)

P = [a b c d]

Q = [a b; c d]

v[i] = trace((P.'\*P)^4)

w[i] = trace((Q.'\*Q)^4)

**end**

std(v)/mean(v), std(w)/mean(w)

**end**

Пример параллельного вычисления 100.000.000 результатов случайного подбрасывания монеты:

nheads = @parallel (+) **for** i=1:100000000

randbit()

**end**

[[3]](#_3._Julia)

# Список использованной литературы

1. [Википедия. Groovy](https://ru.wikipedia.org/wiki/Groovy)
2. [Википедия. Fandom](https://ru.wikipedia.org/wiki/FANTOM)
3. [Википедия. Julia](https://en.wikipedia.org/wiki/Julia)

# Опрос