Основы Julia

Почему Julia?

- Язык разработан специально для научных вычислений
- Богатый набор библиотек для обработки данных, численного моделирования и создания изображений
- Основан на открытых технологиях
- Низкий порог входа
- Удобный синтаксис
- Легко писать эффективные с точки зрения скорости вычислений программы

Особенности Julia

- Интерпретируемый язык со строгой динамической типизацией и встроенной JIT (Just-In-Time) компиляцией
- Интерпретируемый означает, что программы не требуют предварительной компиляции, код можно выполнять по ходу его написания
  - Примеры: Matlab, Python vs C/C++/Fortran
- Строгая типизация означает, что корректность передаваемого типа всегда можно проверить без непосредственного исполнения программы
  - Примеры: Python vs C/C++/Fortran
- Динамическая типизация означает, что тип определяется в процессе выполнения, его явное указание не требуется, тип переменной можно поменять
  - Примеры: Python vs C/C++/Fortran
- JIT-компиляция означает, что в момент первого вызова функции она компилируется с использованием преобразования в C-функцию при помощи LLVM (Low Level Virtual Machine)

    - Примеры: `numba` в Python

- Julia использует Multiple dispatch (множественная генерация методов функции): одна и та же функция компилируется в разные методы в зависимости от используемых типов данных

```
> function mult(x,y)
    x*y
  end
> mult(2, 3)
6
> mult(2.0, 3.0)
6.0
> mult([2, 3]', [1, 4])
14
> mult("H", "i")
"Hi"
```

Julia vs C/C++/Fortran

- Julia проще: легче обучиться, более быстрая разработка, меньше кода
- Сравнимая скорость работы программ
- В Julia встроен автоматический сборщик мусора
- Сложнее писать оптимизированный код
- Сложно создавать скомпилированные в бинарный код приложения

Julia vs Matlab

- Matlab платный, код закрыт
- Язык Matlab заточен под обработку векторов и матриц, написание более общих программ затруднено
- На Matlab сложнее писать эффективные с точки зрения скорости программы
- В Matlab более богатый выбор уже готовых для использования алгоритмов
- У Julia нет сравнимой по удобству использования оболочки

- Matlab значительно лучше документирован

Julia vs Python

- На Python сложнее писать эффективные с точки зрения скорости программы
- К Julia проще подключить библиотеку, написанную на C/C++
- Python изначально разрабатывался не для научных вычислений
- У Python значительно больше коммьюнити
- На Python написано больше кода (но его можно легко подключать к Julia)

Недостатки Julia

- Это молодой язык (первый релиз — 2012 год)
- Относительно небольшое сообщество
- Некоторые вещи или не написаны, или пока сырые (но есть обёртки над практически всеми популярными библиотеками из других языков)
- Часто скудная документация к библиотекам
- Более сложная, чем в Python и Matlab система типов
- Есть сложности с созданием скомпилированных программ

Установка Julia

- С официального сайта: https://julialang.org/downloads/
- Далее: `juliaup`
- Под Linux не рекомендуется пользоваться пакетным менеджером (там обычно старая версия)

- Под Windows рекомендую использовать WSL (Windows Subsystem for Linux)

Использование Julia

Из REPL

- REPL (read, evaluate, print, repeat) — интерактивная оболочка, тут же выполняющая всё, что вводится

```
$ julia
```

```
               _
   _       _ _(_)_     |  Documentation:
https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.11.1 (2024-10-16)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/
release
|__/                   |

julia>
```

- Некоторые полезные комманды:
- `Ctrl-D` — выйти из REPL
- `Ctrl-C` — прервать выполнение команды
- `?` — помощь
- `?<function>` — документация к функции
- `;` — вход в оболочку командной строки
- `Ctrl-L` — очистить экран
- `Ctrl-R`, `Ctrl-S` — поиск назад и вперёд по истории ввода
- "Переменная" `ans` содержит значение, возвращённое предыдущей строкой

Другие способы использования Julia

- Написание программы в файле (обычно с расширением `.jl`) и его запуск из командной строки:

```
$ julia script.jl arg1 arg2
```

- Из IDE (Integrated Development Environment)
  - Рекомендуемый редактор: Visual Studio Code и расширение Julia для него

- Из интерактивных блокнотов Jupyter Notebook (требуется установка пакета `IJulia.jl`)

- Из интерактивных блокнотов Pluto (требуется установка пакета `Pluto.jl`)
  - Главное преимущество: реактивность (автоматическое обновление всех зависимых блоков кода)

Управление пакетами

- Используется специальный встроенный в язык менеджер пакетов `Pkg`

- Менеджер вызывается нажатием `]` в Julia REPL

```
julia>
(@v1.11) pkg>
```

- Основные комманды:
- `add PackageName` — установить пакет
- `remove PackageName` — удалить пакет
- `status` — показать установленные пакеты и их версию
- `update` — обновить пакеты до последней версии
- `activate <name of environment>` — активировать окружение
  - `activate .` — активировать окружение уникальное для текущей папки
- `instantiate` — установить все пакеты для данного окружения (прописываются автоматически или вручную в `Project.toml`)
- `backspace` или `Ctrl-C` — выйти из менеджера

Полезные пакеты

- `Plots.jl`, `Makie.jl` — построение графиков

- `Interpolations.jl`, `SpecialFunctions.jl`, `LinearAlgebra.jl`

- `CSV.jl`, `Tables.jl`, `DataFrames.jl` — работа с данными в табличном виде

- `HDF5.jl`, `JLD2.jl` — работа с данными в формате HDF (Hierarchical Data Format)

- `DifferentialEquation.jl` — решение (в основном обыкновенных) дифференциальных уравнений

- `IJulia.jl` — ядро Julia для Jupyter Notebook

- `FFTW.jl` — обёртка над C-библиотекой `fftw` для быстрого преобразования Фурье

- `Unitful.jl` — типы для данных с единицами измерения

Основы программирования на Julia

Литература

- Официальная документация:
- https://docs.julialang.org/en/v1/

- Обучающие материалы на официальном сайте: https://julialang.org/learning/

- Подсказка по основным функциям: https://cheatsheet.juliadocs.org

- Где искать помощь:
  - https://discourse.julialang.org
  - https://julialang.zulipchat.com
  - LLM

- B. Lauwens, A. Downey. *Think Julia: How to Think Like a Computer Scientist.* https://benlauwens.github.io/ThinkJulia.jl/latest/book.html
- I. Balbaert, A. Salceanu. *Discover Julia, a high-performance language for technical computing.* Packt Publishing. (2019)
- T. Kwong. *Hands-On Design Patterns and Best Practices with Julia.* Packt Publishing. (2020)

Дома

- Установить Julia v1.11.4 https://julialang.org/downloads/
- Установить Visual Studio Code https://code.visualstudio.com/ и расширение Julia для него https://www.julia-vscode.org
- Загрузить файл `julia-basics.jl` и построчно позапускать его в Visual Studio Code
- Написать функцию `std_dev(x)`, которая возвращает стандартное отклонение для всех элементов вектора `x`:

$$\sigma = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 \right)^{\frac{1}{2}}$$

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$