

## **Аннотированный список по языку программирования Julia:**

### **1. Beginning Julia Programming For Engineers and Scientists**

*Автор:* Sandeep Nagar

*Ссылка на источник:*

Beginning Julia Programming. [Электронный ресурс].

URL: <https://www.programmer-books.com/wp-content/uploads/2019/04/Beginning-Julia-Programming.pdf> (дата обращения: 12.09.2024).

Эта книга представляет язык программирования Julia с уклоном в его применение в инженерии и науках. Авторы описывают основные концепции Julia, включая синтаксис, структуры данных и основные библиотеки, демонстрируя, как создавать эффективные программные решения для инженерных задач. Книга включает практические примеры и упражнения для применения полученных знаний на практике. Также обсуждаются темы визуализации данных и работы с массивами, что делает ее полезной для специалистов, занимающихся количественными данными.

### **2. Getting Started with Julia Programming**

*Автор:* Ivo Balbaert

*Ссылка на источник:*

Документ ВКонтакте. [Электронный ресурс].

URL: [https://vk.com/doc7368217\\_499247271?hash=kC2YmaiSmbWEwIIzgJS0U9KfIU4Xd6lsejN3hFpFFnk&dl=qdjlFyapd5jGrbwTmkkdQMUhO8jQ7Uwi4eOupawqTXL](https://vk.com/doc7368217_499247271?hash=kC2YmaiSmbWEwIIzgJS0U9KfIU4Xd6lsejN3hFpFFnk&dl=qdjlFyapd5jGrbwTmkkdQMUhO8jQ7Uwi4eOupawqTXL) (дата обращения: 12.09.2024).

Этот ресурс предназначен для новичков, желающих быстро освоить язык программирования Julia. Книга предлагает структурированный подход к изучению, начиная с установки и настройки среды разработки. В ней объясняется, как использовать язык для решения простых задач, представлены основные конструкции и функции. Выделены преимущества Julia по сравнению с другими языками программирования, особенно в контексте производительности и работы с числовыми данными. Включены практические примеры и советы по оптимизации кода.

### **3. Julia 1.0 Programming Complete Reference Guide**

*Автор:* Ivo Balbaert & Adrian Salceanu

*Ссылка на источник:*

Документ ВКонтакте. [Электронный ресурс].  
URL: [https://vk.com/doc409016625\\_612990743?hash=Uq5fCGhoZpUGJi9nLzOjW7AxzdB2Z2Zp04S21Z6d4wz&dl=zgNqBTgznd2Em3ZVKciabuKePhKZjgu2tS7zCDzoLWs](https://vk.com/doc409016625_612990743?hash=Uq5fCGhoZpUGJi9nLzOjW7AxzdB2Z2Zp04S21Z6d4wz&dl=zgNqBTgznd2Em3ZVKciabuKePhKZjgu2tS7zCDzoLWs) (дата обращения: 12.09.2024).

Это полное руководство по языку программирования Julia, охватывающее все аспекты работы с последней стабильной версией 1.0. Книга служит как справочник для опытных пользователей, так и учебным пособием для новичков. В ней подробно описаны основные функции языка, синтаксис и особенности работы с библиотеками. Рассматриваются расширенные темы, такие как метапрограммирование, параллельные вычисления и работа с внешними библиотеками. Приводятся примеры использования Julia в научных вычислениях, что делает книгу ценным ресурсом.

#### **4. Краткое описание языка программирования Julia и некоторые примеры его использования**

*Автор:* Белов Г.В.

*Ссылка на источник:*

Brief description of Julia language. [Электронный ресурс].  
URL: [http://ihed.ras.ru/~thermo/Julia/Brief description of Julia language.pdf](http://ihed.ras.ru/~thermo/Julia/Brief%20description%20of%20Julia%20language.pdf).  
(дата обращения: 13.09.2024).

В этом источнике представлено краткое введение в язык программирования Julia, его ключевые особенности и достоинства. Автор акцентирует внимание на производительности языка, удобстве работы с массивами и числовыми данными. В книге содержатся примеры, иллюстрирующие основные конструкторы языка и его возможности в решении научных и инженерных задач. Также рассматриваются популярные библиотеки и инструменты, которые облегчают работу программистов, использующих Julia для анализа данных и численных расчетов.

#### **5. Осваиваем язык Julia - Малькольм Шеррингтон, 2017**

*Автор:* Малькольм Шеррингтон

*Ссылка на источник:*

Документ ВКонтакте. [Электронный ресурс].  
URL: [https://vk.com/doc247540843\\_498786958?hash=WkPI9bAR818o3csibdSuk9BEBKdgPLh2QptnLnnqtLT&dl=nj6ETopZmZqRPTzvvOXFz4JRz03Zni0ZtU2OPMy1LRw](https://vk.com/doc247540843_498786958?hash=WkPI9bAR818o3csibdSuk9BEBKdgPLh2QptnLnnqtLT&dl=nj6ETopZmZqRPTzvvOXFz4JRz03Zni0ZtU2OPMy1LRw) (дата обращения: 13.09.2024).

Книга является доступным руководством для изучения языка Julia, в котором рассматриваются как базовые, так и более сложные аспекты программирования. Малькольм Шеррингтон делится практическим

опытом использования Julia в различных приложениях, акцентируя внимание на его актуальности для научных исследований и приложений в области обработки данных. Книга содержит множество примеров и задач, позволяющих читателю поэтапно освоить язык, а также освещает темы, такие как работа с графиками и машинным обучением. Это издание станет незаменимым помощником как для начинающих, так и для опытных программистов, стремящихся расширить свои знания о Julia.

## 6. Julia 1.10 Documentation

*Автор:* не указан

*Ссылка на источник:*

Julia Documentation. [Электронный ресурс].

URL: <https://docs.julialang.org/en/v1/> (дата обращения: 13.09.2024).

Сайт представляет официальную документацию языка программирования Julia на английском языке. Здесь можно найти полезные материалы, инструкции, примеры кода и руководства для работы с этим языком. Julia - высокопроизводительный динамический язык программирования, который широко используется для научных вычислений, анализа данных и машинного обучения. Сайт предлагает различные возможности для изучения и использования Julia, помогая программистам расширить свои знания и навыки в области разработки программного обеспечения.

### Примеры кода на языке Julia

#### 1. Код нахождения суммы элементов вектора

```
1  # Определяем функцию sum_vec, которая принимает вектор v как аргумент
2  function sum_vec(v)
3      # Инициализируем переменную total, которая будет хранить сумму
4      total = 0
5
6      # Проходим по каждому элементу вектора v
7      for i in v
8          # Добавляем текущий элемент к накопленной сумме total
9          total += i
10     end
11
12     # Возвращаем итоговую сумму элементов
13     return total
14 end
15
16 # Создаем пример вектора
17 vec = [3, 4, 5]
18
19 # Вызываем функцию sum_vec и предоставляем ей вектор vec
20 result = sum_vec(vec)
21
22 # Выводим результат на экран
23 println("Сумма элементов вектора: ", result)
```

```
▼ Interpret
Сумма элементов вектора: 12
julia>
```

Сначала мы создаем функцию `sum_vec`, которая принимает один аргумент - вектор `v`. Инициализируем переменную `total` значением 0 для накопления суммы. Используем цикл `for`, чтобы пройтись по каждому элементу вектора `v` и добавлять его к `total`. После завершения цикла функция возвращает итоговую сумму элементов. Далее мы создаем пример вектора `vec` с числами: 3, 4, 5.

Вызываем функцию `sum_vec`, передавая ей наш вектор, и сохраняем результат в переменную `result`. Выводим результат на экран с помощью функции `println`.

## 2. Решение системы линейных уравнений

```
main.jl x +
main.jl
1 # Решение системы линейных уравнений wx = D
2 using LinearAlgebra
3
4 w = [3.0 2.0; 1.0 2.0] # Коэффициенты системы
5 D = [5.0; 4.0]         # Свободные члены
6
7 x = w \ D # Решение системы уравнений
8 println("Решение системы: ", x) # Вывод: [0.5, 1.75]
9
10
```

```
▼ Interpret
Решение системы: [0.5, 1.75]
julia>
```

Импорт библиотеки: В начале программы подключается модуль `LinearAlgebra`, который предоставляет функции для работы с линейной алгеброй.

Определение матрицы и вектора:

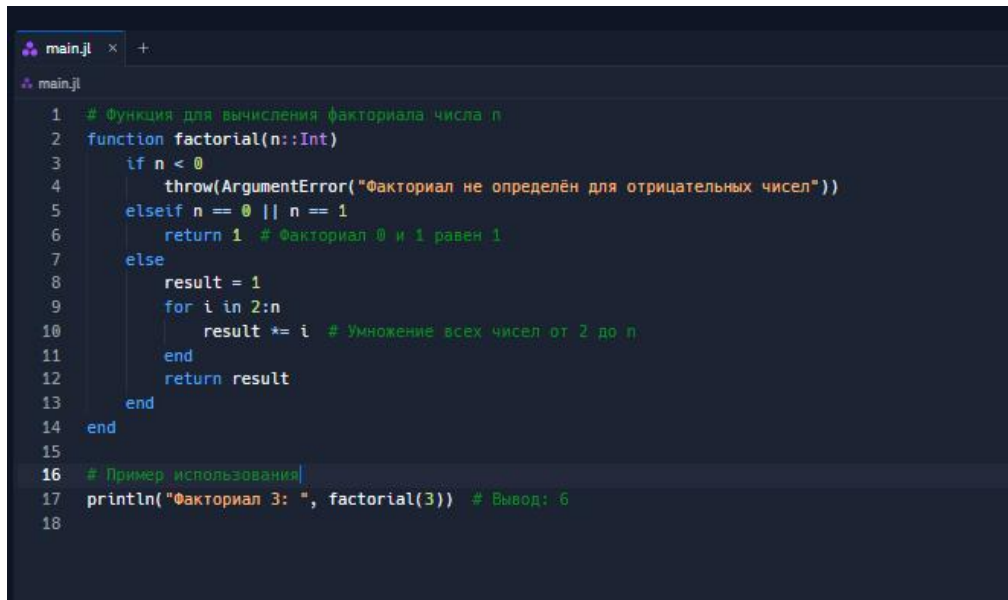
- `w` — это матрица коэффициентов, содержащая значения, которые определяют систему уравнений.

- `D` — это вектор свободных членов, представляющий результаты, соответствующие каждому уравнению.

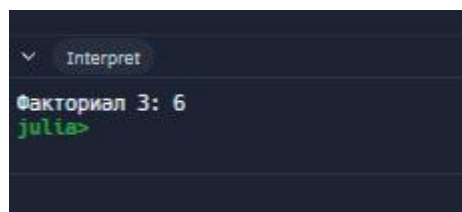
Решение системы: Используется оператор \, который позволяет найти вектор решений, удовлетворяющий системе уравнений.

Вывод результата: Программа выводит найденное решение на экран, которое в данном случае равно [0.5, 1.75].

### 3. Функция для вычисления факториала



```
main.jl x +
main.jl
1 # Функция для вычисления факториала числа n
2 function factorial(n::Int)
3     if n < 0
4         throw(ArgumentError("Факториал не определён для отрицательных чисел"))
5     elseif n == 0 || n == 1
6         return 1 # Факториал 0 и 1 равен 1
7     else
8         result = 1
9         for i in 2:n
10             result *= i # Умножение всех чисел от 2 до n
11         end
12         return result
13     end
14 end
15
16 # Пример использования
17 println("Факториал 3: ", factorial(3)) # Вывод: 6
18
```



```
Interpret
Факториал 3: 6
julia>
```

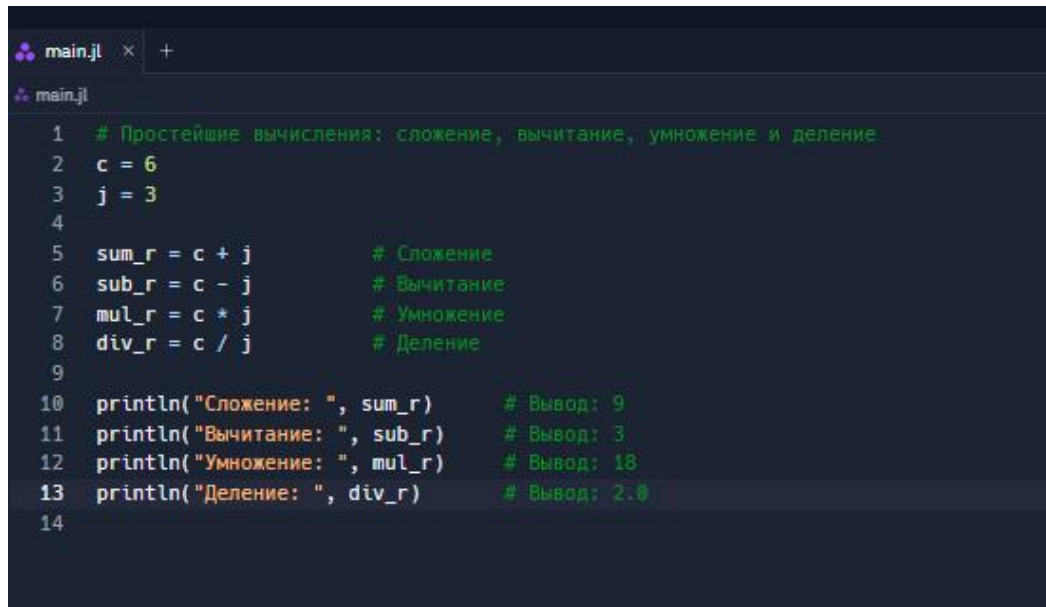
Функция `factorial(n::Int)`:

- Принимает одно целое число `n` в качестве аргумента.
- Проверяет, является ли `n` отрицательным. Если да, выбрасывается ошибка с сообщением о том, что факториал для отрицательных чисел не определён.
- Если `n` равно 0 или 1, функция возвращает 1, так как факториал этих чисел равен 1.
- Для положительных `n` функция использует цикл для умножения всех целых чисел от 2 до `n`, чтобы вычислить факториал.

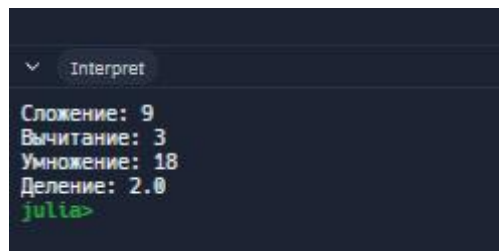
В конце программы приведён пример вызова функции для вычисления факториала числа 3, результатом которого является 6.

Таким образом, программа демонстрирует простой способ вычисления факториала с обработкой ошибок и использованием циклов.

#### 4. Простейшие вычисления



```
main.jl x +
main.jl
1 # Простейшие вычисления: сложение, вычитание, умножение и деление
2 c = 6
3 j = 3
4
5 sum_r = c + j          # Сложение
6 sub_r = c - j          # Вычитание
7 mul_r = c * j          # Умножение
8 div_r = c / j          # Деление
9
10 println("Сложение: ", sum_r)    # Вывод: 9
11 println("Вычитание: ", sub_r)   # Вывод: 3
12 println("Умножение: ", mul_r)   # Вывод: 18
13 println("Деление: ", div_r)     # Вывод: 2.0
14
```



```
▼ Interpret
Сложение: 9
Вычитание: 3
Умножение: 18
Деление: 2.0
julia>
```

Программа определяет две переменные  $c$  и  $j$  со значениями 6 и 3 соответственно.

Арифметические операции:

- Сложение: вычисляется сумма  $c$  и  $j$ .
- Вычитание: вычисляется разность  $c$  и  $j$ .
- Умножение: вычисляется произведение  $c$  и  $j$ .
- Деление: вычисляется частное  $c$  и  $j$ .

Результаты всех операций выводятся на экран с поясняющими комментариями.

## 5. Работа с массивами

```
main.jl x +
main.j Close tab
1 # Работа с массивами
2 v = [1, 2, 3, 4]           # Создание массива
3
4 v[4] = 13                 # Изменение четвертого элемента массива
5 push!(v, 5)               # Добавление нового элемента в массив
6
7 println("Массив: ", v)    # Вывод: [1, 2, 3, 13, 5]
8 println("Длина массива: ", length(v)) # Вывод: 5
9
```

```
Interpret
Массив: [1, 2, 3, 13, 5]
Длина массива: 5
julia>
```

Данная программа демонстрирует работу с массивами в языке программирования Julia.

Инициализируется массив `v` с элементами от 1 до 4. Четвертый элемент массива изменяется на 13. В массив добавляется новый элемент со значением 5. Программа выводит обновленный массив и его длину на экран.

Таким образом, программа иллюстрирует основные операции с массивами, включая создание, изменение и добавление элементов.

## 6. Работа с циклами

```
main.jl x +
main.jl
1 # Пример работы с циклами: вывод четных чисел от 1 до 5
2 for n in 1:5
3     if n % 2 == 0           # Проверка на четность
4         println(n)         # Вывод четного числа
5     end
6 end
7
```

```
Interpret
2
4
julia>
```

Данная программа на языке Julia демонстрирует использование цикла для вывода четных чисел в диапазоне от 1 до 5.

Цикл `for`: Проходит по всем целым числам от 1 до 5. Затем для каждого числа выполняется проверка, является ли оно четным (остаток от деления на 2 равен 0). Если число четное, оно выводится на экран с помощью функции `println`.