

Язык программирования Julia

Аннотированные статьи и ресурсы:

1. [Официальная документация Julia:](#)

Этот ресурс является основным источником информации для изучения языка Julia. Он содержит подробные руководства по синтаксису, встроенным функциям и структурам данных.

2. [Руководство для начинающих от JuliaLang:](#)

Это руководство разработано специально для новичков в программировании и языке Julia. Оно охватывает основы языка, включая переменные, массивы, функции и модули.

3. [Язык программирования математических вычислений Julia. Базовое руководство:](#)

Учебно-методическое пособие представляет собой базовое руководство по языку Julia. Пособие содержит сведения по установке интерпретатора языка Julia, принципах работы, основных конструкциях и возможностях языка.

4. [Язык Julia как инструмент исследователя:](#)

Пособие знакомит читателей с новым языком программирования Julia. Показывает возможности и особенности, а также на простых примерах иллюстрируются основные идеи в реализации языка Julia.

5. [Научное программирование на языке Julia:](#)

Презентация содержит большое количество полезной информации о языке программирования Julia. Описаны особенности языка, средства разработки. Также представлен синтаксис языка и программы.

6. [Julia. Язык программирования. Быстрый старт:](#)

Книга по языку Julia с краткими и понятными пояснениями по быстрому старту. Подойдет для начинающих.

Примеры решения задач на языке Julia:

1. Вычисление факториала:

```
examples.jl > ...
1  function factorial(n)
2      if n == 0 || n == 1
3          return 1
4      else
5          return n * factorial(n - 1)
6      end
7  end
8
9  println("Факториал 5: ", factorial(5))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

Факториал 5: 120

julia>

Эта функция рекурсивно вычисляет факториал числа n . Если n равно 0 или 1, то возвращается 1. В противном случае результат получается умножением n на факториал $(n-1)$.

2. Поиск наибольшего общего делителя двух чисел:

```
examples.jl > ...
1  function gcd(a, b)
2      while b != 0
3          t = b
4          b = a % b
5          a = t
6      end
7      return a
8  end
9
10 println("НОД для 48 и 60: ", gcd(48, 60))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

НОД для 48 и 60: 12

julia>

Алгоритм Евклида используется для поиска наибольшего общего делителя (НОД) чисел a и b . Пока b не станет равным нулю, переменная t сохраняет текущее значение b , затем b становится остатком от деления $a \% b$, а a обновляется до значения t . Когда b станет равной нулю, a будет равен НОД.

3. Нахождение наименьшего положительного корня квадратного уравнения:

```
examples.jl > ...
1 function solve_quadratic(a, b, c)
2     discriminant = b^2 - 4*a*c
3
4     if discriminant >= 0
5         root1 = (-b + sqrt(discriminant)) / (2*a)
6         root2 = (-b - sqrt(discriminant)) / (2*a)
7         println("Корни: $root1, $root2")
8     else
9         println("Уравнения нет действительных корней.")
10    end
11 end
12
13 solve_quadratic(1, 5, 6)
```

Корни: -2.0, -3.0

julia>

Функция решает квадратное уравнение вида $ax^2+bx+c=0$. Она рассчитывает дискриминант b^2-4ac и проверяет его на неотрицательность. Если дискриминант больше или равен нулю, вычисляются два корня уравнения. Если дискриминант меньше нуля, сообщается, что у уравнения нет действительных корней.

4. Нахождение наименьшего положительного корня квадратного уравнения:

```
examples.jl X
examples.jl > ...
1 function solve_quadratic(a, b, c)
2     discriminant = b^2 - 4*a*c
3
4     if discriminant >= 0
5         root1 = (-b + sqrt(discriminant)) / (2*a)
6         root2 = (-b - sqrt(discriminant)) / (2*a)
7         println("Корни: $root1, $root2")
8     else
9         println("Уравнения нет действительных корней.")
10    end
11 end
12
13 solve_quadratic(1, 5, 6)
```

Корни: -2.0, -3.0

julia>

Функция решает квадратное уравнение вида $ax^2+bx+c=0$. Она рассчитывает дискриминант b^2-4ac и проверяет его на неотрицательность. Если дискриминант больше или равен нулю, вычисляются два корня уравнения. Если дискриминант меньше нуля, сообщается, что у уравнения нет действительных корней.

5. Работа с массивами:

```
examples.jl X
examples.jl > ...
1  function swap!(arr, i, j)
2      temp = arr[i]
3      arr[i] = arr[j]
4      arr[j] = temp
5  end
6
7  arr = [1, 2, 3, 4, 5]
8  push!(arr, 6)
9  popfirst!(arr)
10 length(arr)
11 arr[2] = 10
12 swap!(arr, 3, 4)
13 in(10, arr)
14 println(arr)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[2, 10, 5, 4, 6]

julia>
```

Создаем функцию `swap` и начальный массив. Добавляем новый элемент в конец массива. Удаляем первый элемент. Проверяем длину массива. Заменяем второй элемент на 10. Меняем местами 3 и 4 элементы. Ищем наличие элемента в массиве. Выводим массив.