

## Язык программирования Julia

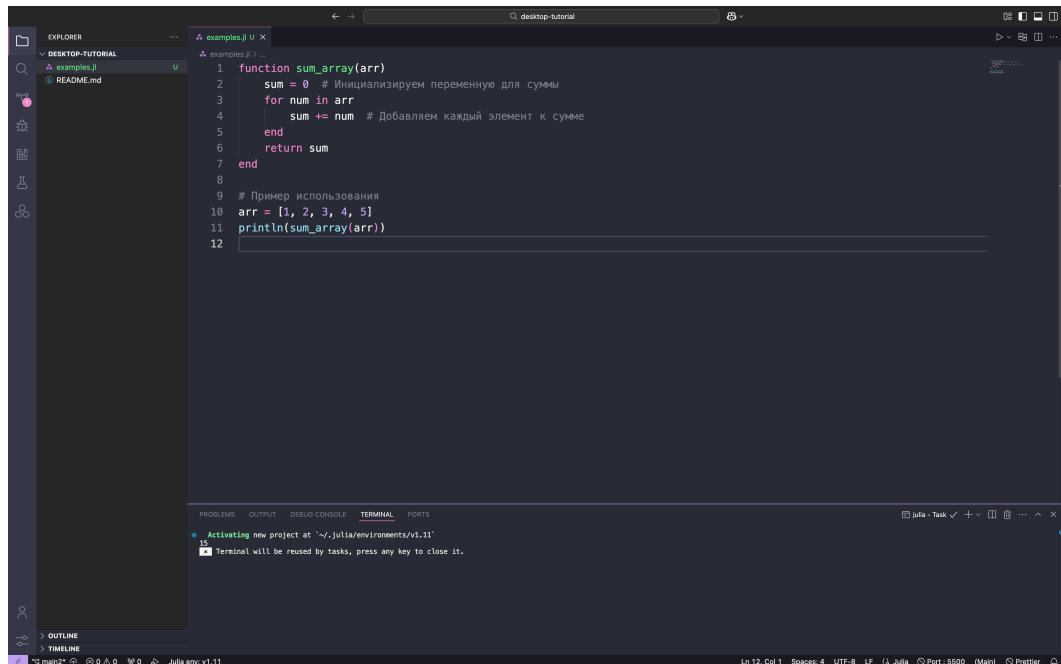
**Julia** — это динамический, высокопроизводительный язык программирования, предназначенный для численных вычислений и научных исследований. Он сочетает удобство Python и скорость C.

### Аннотированные статьи и ресурсы по языку программирования Julia

- Julia 1.10 Documentation // URL: <https://docs.julialang.org/en/v1/> // Официальная документация по языку программирования Julia
- Язык Julia как инструмент исследователя // URL: [https://cmp.phys.msu.ru/sites/default/files/VA\\_Antonyk\\_Julia\\_2019.pdf?utm\\_source=chatgpt.com](https://cmp.phys.msu.ru/sites/default/files/VA_Antonyk_Julia_2019.pdf?utm_source=chatgpt.com) // Это учебное пособие знакомит читателей с языком Julia, позиционируемым как язык для научного программирования. Оно описывает особенности и возможности языка, а также его применение в научных и технических задачах.
- Julia. Язык программирования. Быстрый старт // URL: [https://www.litres.ru/book/vadim-nikitin-32700223/julia-yazyk-programmirovaniya-bystryy-start-69596290/?utm\\_source=chatgpt.com](https://www.litres.ru/book/vadim-nikitin-32700223/julia-yazyk-programmirovaniya-bystryy-start-69596290/?utm_source=chatgpt.com) // Книга Вадима Никитина предлагает быстрое введение в язык Julia, охватывая базовые концепции и предоставляя практические примеры для начинающих. Она поможет быстро освоить основы и приступить к написанию собственных программ на Julia.
- Возможности языка Julia для обработки статистических данных // URL: [https://journals.rudn.ru/miph/article/view/34459/ru\\_RU?utm\\_source=chatgpt.com](https://journals.rudn.ru/miph/article/view/34459/ru_RU?utm_source=chatgpt.com) // В этой статье рассматривается применение языка Julia в области математической статистики. Авторы подробно описывают процесс установки и настройки программного окружения, а также анализируют доступные библиотеки для статистической обработки данных.
- Научное программирование на языке Julia // URL: [https://exponenta.ru/storage/app/media/Conf\\_2023/Презентации\\_с\\_конференции\\_2023/Моделирование%20в%20инженерном%20деле\\_5.04.2023/khirulin\\_kamil.pdf](https://exponenta.ru/storage/app/media/Conf_2023/Презентации_с_конференции_2023/Моделирование%20в%20инженерном%20деле_5.04.2023/khirulin_kamil.pdf) //

# Примеры решения различных задач

## 1. Сумма элементов массива

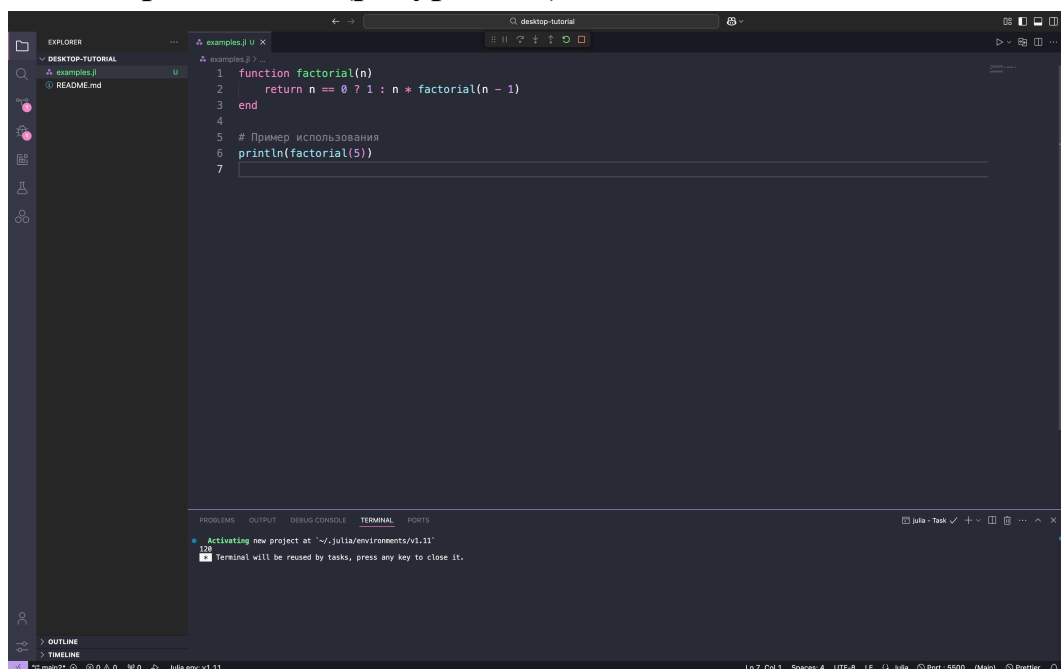


The screenshot shows the Visual Studio Code editor with a file named `examples.jl` open. The code defines a function `sum_array(arr)` that calculates the sum of elements in an array. It includes a comment in Russian: "# Инициализируем переменную для суммы" (Initialize the variable for the sum). The function uses a `for` loop to iterate over the array and accumulate the sum. Below the function, there is an example of its usage: creating an array `arr = [1, 2, 3, 4, 5]` and printing the result of `sum_array(arr)`. The terminal at the bottom shows the message: "Activating new project at '~/.julia/environments/v1.11'" and "Terminal will be reused by tasks, press any key to close it."

```
1 function sum_array(arr)
2     sum = 0 # Инициализируем переменную для суммы
3     for num in arr
4         sum += num # Добавляем каждый элемент к сумме
5     end
6     return sum
7 end
8
9 # Пример использования
10 arr = [1, 2, 3, 4, 5]
11 println(sum_array(arr))
12
```

Функция вычисляет сумму всех элементов массива.

## 2. Факториал числа (рекурсивно)

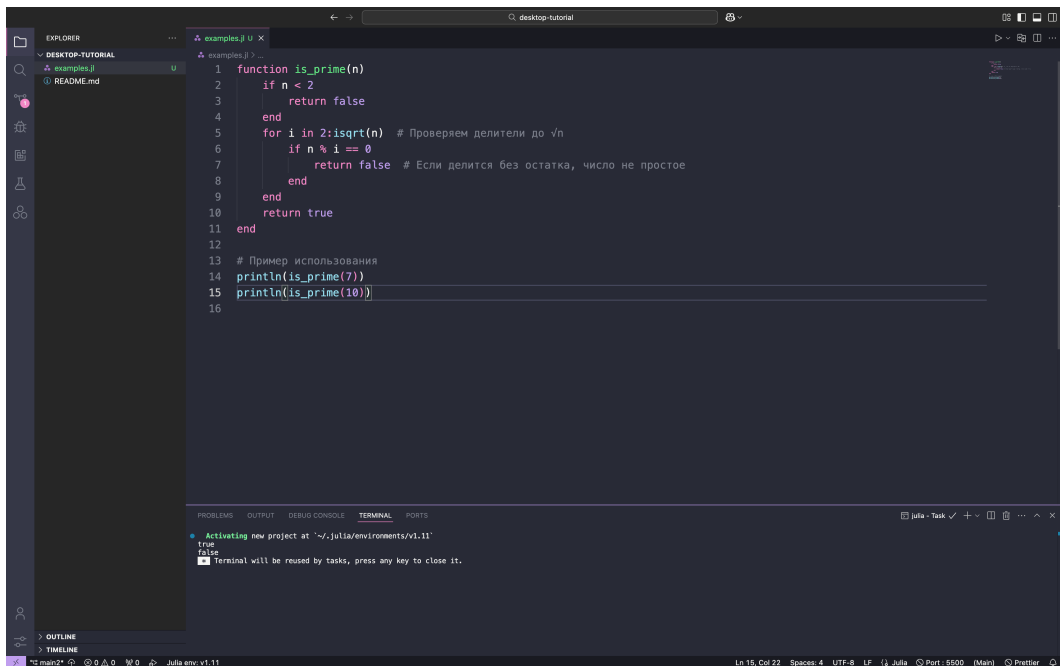


The screenshot shows the Visual Studio Code editor with a file named `examples.jl` open. The code defines a recursive function `factorial(n)` to calculate the factorial of a number `n`. It includes a base case: `return n == 0 ? 1 : n * factorial(n - 1)`. Below the function, there is an example of its usage: printing the result of `factorial(5)`. The terminal at the bottom shows the message: "Activating new project at '~/.julia/environments/v1.11'" and "Terminal will be reused by tasks, press any key to close it."

```
1 function factorial(n)
2     return n == 0 ? 1 : n * factorial(n - 1)
3 end
4
5 # Пример использования
6 println(factorial(5))
7
```

Функция вычисляет факториал числа `n`, используя рекурсию.

## 3. Проверка, является ли число простым

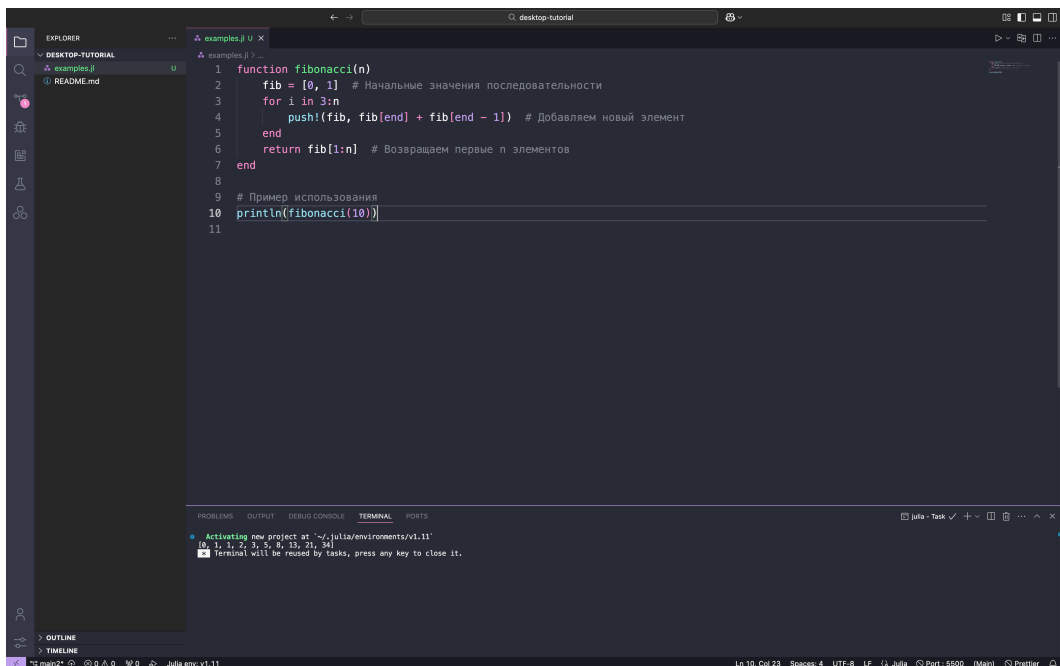


The screenshot shows a VS Code editor with a file named `examples.jl` open. The code defines a function `is_prime(n)` that checks if a number `n` is prime. It uses a loop to check divisors from 2 to  $\sqrt{n}$ . Comments in Russian explain the logic. Below the function, there is a usage example: `println(is_prime(7))` and `println(is_prime(10))`. The terminal at the bottom shows the message: "Activating new project at '~/.julia/environments/v1.11'" and "true".

```
1 function is_prime(n)
2     if n < 2
3         return false
4     end
5     for i in 2:isqrt(n) # Проверяем делители до √n
6         if n % i == 0
7             return false # Если делится без остатка, число не простое
8         end
9     end
10    return true
11 end
12
13 # Пример использования
14 println(is_prime(7))
15 println(is_prime(10))
16
```

Функция проверяет, является ли число `n` простым.

## 4. Генерация чисел Фибоначчи



The screenshot shows a VS Code editor with a file named `examples.jl` open. The code defines a function `fibonacci(n)` that generates the first `n` Fibonacci numbers. It uses an array `fib` and a loop to calculate the sequence. Comments in Russian explain the logic. Below the function, there is a usage example: `println(fibonacci(10))`. The terminal at the bottom shows the message: "Activating new project at '~/.julia/environments/v1.11'" and the output: "(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)".

```
1 function fibonacci(n)
2     fib = [0, 1] # Начальные значения последовательности
3     for i in 3:n
4         push!(fib, fib[end] + fib[end - 1]) # Добавляем новый элемент
5     end
6     return fib[1:n] # Возвращаем первые n элементов
7 end
8
9 # Пример использования
10 println(fibonacci(10))
11
```

Функция генерирует первые `n` чисел Фибоначчи.

## 5. Поиск максимального элемента в массиве

The screenshot shows a Julia IDE window with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor contains a Julia script named `examples.jl` with the following content:

```
1 function max_element(arr)
2     max_val = arr[1] # Берём первый элемент за максимальный
3     for num in arr
4         if num > max_val
5             max_val = num # Обновляем максимум
6         end
7     end
8     return max_val
9 end
10
11 # Пример использования
12 arr = [3, 7, 2, 9, 1, 5]
13 println(max_element(arr))
14
```

The terminal at the bottom shows the output of the script:

```
Activating new project at '~/.julia/environments/v1.11'
9
Terminal will be reused by tasks, press any key to close it.
```

Функция находит наибольший элемент в массиве.