Данные и переменные в Julia

Перцев Дмитрий, Никита Швачко

24 октября 2025 г.

Содержание

Что такое переменная?

- Переменная — именованная область памяти для хранения данных
- В Julia переменные создаются при первом присваивании:

```
x = 10
name = "Julia"
pi_value = 3.14159
```

- Динамическая типизация — тип определяется автоматически
- Можно перезаписывать значения разных типов:

```
x = 10 # Int64
x = 3.14 # Float64
```

Правила именования

Допустимые имена

- Латинские буквы (a-z, A-Z)
- Цифры (0-9), кроме первого символа
- Символы Unicode
- Подчеркивания (_)

Примеры

```
var1 = 5
my_variable = "text"
```

$\delta = 0.01$

Базовые типы данных

Тип	Пример
Целые числа	`Int64, Int32`
Числа с плавающей точкой	`Float64`
Логический тип	`Bool`
Строки	`String`
Символы	`Char`

Примеры разных типов:

```
integer_var = 42
float_var = 3.14
bool_var = true
string_var = "Hello Julia"
char_var = 'J'
```

Множественное присваивание

Одновременное присваивание

```
a, b, c = 1, 2, 3
x = y = z = 0 # Все переменные получают значение 0
```

Обмен значений

```
a, b = b, a # Классический обмен значений
```

Динамическая типизация

Особенности Julia

- Типы определяются во время выполнения
- Возможна смена типа переменной
- Поддержка полиморфизма

```
x = 10 # Int64
println(typeof(x)) # Выведет Int64

x = "Теперь строка"# String
println(typeof(x)) # Выведет String
```

Объявление констант

Ключевое слово const

```
const PI = 3.14159
const SPEED_OF_LIGHT = 299792458
```

Важно

- Константы нельзя изменять
- Попытка изменения вызовет предупреждение
- Рекомендуется для значений, которые не должны меняться

Работа со строками

Основные операции

```
s1 = "Hello"
s2 = "World"
concat = s1 * s2 # Конкатенация
length = sizeof(s1) # Длина строки
```

Интерполяция строк

```
name = "Анна"
age = 25
message = "Привет, $name! Тебе $age лет."
```

Базовые структуры данных

Массивы

```
arr = [1, 2, 3, 4, 5] # Вектор
matrix = [1 2; 3 4] # Матрица
multi_type = [1, "hello"] # Разнотипный массив
```

Операции с массивами

```
push!(arr, 6) # Добавление элемента
pop!(arr) # Удаление последнего элемента
length(arr) # Длина массива
```

Словари (ассоциативные массивы)

Создание и использование

```
person = Dict("name-> "Иван"age-> 30)
person["name"] # Доступ по ключу
person["city"] = "Москва"# Добавление элемента
```

Полезные методы

```
keys(person) # Все ключи
values(person) # Все значения
haskey(person, "age") # Проверка наличия ключа
```

Что такое структуры?

Определение структур

Структуры — пользовательские составные типы данных, которые группируют связанные данные

Пример определения структуры

```
struct Point
 x::Float64
 y::Float64
end
```

Особенности

- По умолчанию неизменяемые (immutable)
- Высокая производительность
- Поддержка методов и наследования

Создание и использование структур

Создание экземпляров структур

```
p = Point(3.0, 4.0) # Создание точки
println(p.x) # Доступ к полю x
println(p.y) # Доступ к полю y
```

Структура с разными типами данных

```
struct Person
 name::String
 age::Int
 email::String
end
```

Изменяемые структуры

Ключевое слово mutable

```
mutable struct MutablePoint
 x::Float64
 y::Float64
end
```

Изменение полей

```
mp = MutablePoint(1.0, 2.0)
mp.x = 5.0 # Теперь можно изменять!
mp.y = 10.0
```

Производительность

- Неизменяемые структуры быстрее
- Изменяемые структуры удобнее для часто изменяемых данных

Ключевые моменты

- Простое объявление переменных при присваивании
- Динамическая типизация с высокой производительностью
- Поддержка Unicode в именах переменных
- Гибкие структуры данных
- Мощные возможности для работы со строками
- Эффективные массивы и коллекции
- Структуры для создания пользовательских типов
- Неизменяемые и изменяемые структуры
- Методы и конструкторы для структур