

Числа

3	целое число
0x1F	целое число
3.0	число с плавающей запятой

Кортежи

{1,2,3} # кортеж
elem({1, 2, 3}, 0) #=> 1

Кортежи, хранятся в памяти последовательно.

Получить доступ к элементу кортежа мы можем с помощью функции "elem"

Атомы

:hello # атом

Атомы, которые являются нечисловыми константами. Они начинаются с символа :

Бинарные данные

<<1,2,3>>

Оператор "?"

?a #=> 97

Оператор "?" возвращает целое число, соответствующее данному символу.

Карты (Maps)

Maps are key-value pairs:

```
genders = %{"david" => "male", "gillian" => "female"}
genders["david"] #=> "male"
```

Maps with atom keys:

```
genders = %{david: "male", gillian: "female"}
genders.gillian #=> "female"
```

Строки

"- Elixir-строка (заключена в двойные кавычки)
hel-
lo"

Строки (cont)

'hello' Erlang-строка (в одинарные кавычки)

Все строки представлены в кодировке UTF-8:

"привет" #=> "привет"

Elixir-строки / Erlang-строки

<<?a, ?b, ?c>> #=> "abc"

Elixir-строки являются бинарными данными

[?a, ?b, ?c] #=> 'abc'

Erlang-строка — это на самом деле список

Конкатенация строк

"<>" - для объединения бин. данных (и Elixir-строк)

<<1,2,3>> <> <<4,5>> #=> <<1,2,3,4,5>>

"hello " <> "world" #=> "hello world"

"++" - для объединения списков (и Erlang-строк)

[1,2,3] ++ [4,5] #=> [1,2,3,4,5]

'hello ' ++ 'world' #=> 'hello world'

Математические операции

1 + 1 # 10 - 5 # 5 * 2 # 10 / 2

div(10, 2) #=> целочисленное деление
5

rem(10, 3) #=> 1 получение остатка от деления

В Elixir оператор "/" всегда возвращает число с плавающей запятой.

Relaxed boolean

1 || true #=> 1 false && 1 #=> false

nil && 20 #=> nil !true #=> false

a || b gives true if a is true. Otherwise, it gives b.

a && b gives false if a is false. Otherwise, it gives b.

!a gives false if a is true. Otherwise, it gives true.

Списки (Lists)

[1,2,3] Списки, реализованы как связанные списки

We can access the **head and tail** of a list as follows:

[head | tail] = [1,2,3]

head #=> 1

tail #=> [2,3]

Диапазоны (Ranges)

Ranges are represented as "start..end" (включительно): 1..10 #=> 1..10

Сопоставление с образцом применимо и для ranges:

lower..upper = 1..10

[lower, upper] #=> [1, 10]

Операторы сравнения

a===b strict equality (so 1 === 1.0 is false)

a!==b strict inequality (so 1 !== 1.0 is true)

a==b equality (so 1 == 1.0 is true)

a!=b inequality (so 1 != 1.0 is false)

a>b greater than the other (so 2>1 is true)

a>=b greater than or equal (so 2>=1 is true)

a<b less than the other (so 1<2 is true)

a<=b less than or equal (so 1<=2 is true)

Elixir позволяет сравнивать значения разных типов:

1 < :hello #=> true

Правило: число < атом < ссылка < функция < порт < процесс < кортеж < список < строка

Булевые: or, and, not.

true and true #=> true

false or true #=> true

1 and true #=> ** (BadBooleanError)

В качестве первого аргумента эти операторы ожидают булево значение.