

Язык программирования Elixir



- **Введение в типы данных Elixir**

Выполнили Фролов Иван и Ткачев Михаил, гр. 5030102/20202

Числа

integer = 42

float = 3.14

Примитивные типы данных

Атомы (символы)

atom = :hello

boolean_true = true

boolean_false = false

Строки и charlists

string = "Hello"

charlist = 'world'

Битовые строки и бинарные данные

binary = <<1, 2, 3>>

- Вывод в консоль:
- iex> integer = 42
- 42
- iex> :hello
- :hello
- iex> "Hello"



elixir

Пользовательские типы данных

- Кортежи (Tuples)
- `tuple = {:ok, "result", 200}`
- Списки
- `list = [1, 2, 3, 4]`
- Карты (Maps)
- `map = %{name: "John", age: 25}`
- Структуры (Structs)
- `defmodule User do`
- `defstruct name: "", age: 0`
- Вывод в консоль:
- `iex> tuple = {:ok, "result"}`
- `{:ok, "result"}`
- `iex> %User{name: "Alice"}`
- `%User{name: "Alice", age: 0}`



Операции над типами

Арифметические операции

`sum = 5 + 3.2`

8.2

`concat = "Hello " <> "World"`

Hello World

Операции со списками

`combined_list = [1, 2] ++ [3, 4]`

[1, 2, 3, 4]

`head = hd([1, 2, 3])`

1

`tail = tl([1, 2, 3])`

[2, 3]

Операции с картами

`updated_map = Map.put(map, :city, "Berlin")`



Преобразование типов

Явное преобразование

```
number_string = "123"
```

```
{number, _} = Integer.parse(number_string)
```

```
float_number = Float.parse("3.14")
```

Неявное преобразование

```
binary_to_string = <<104, 101, 108, 108, 111>> "hello"
```

Функции преобразования

```
string_to_integer = String.to_integer("42")
```

```
integer_to_string = Integer.to_string(42)
```

```
atom_to_string = Atom.to_string(:hello)
```



Сравнение типов

Строгое сравнение

```
strict_equal = 1 === 1.0
```

```
strict_not_equal = 1 !== 1.0
```

```
iex> 1 === 1.0
```

```
false
```

```
iex> 1 == 1.0
```

```
true
```

```
iex> 1 < :atom
```

```
true
```

Нестрогое сравнение

```
loose_equal = 1 == 1.0
```

```
loose_not_equal = 1 != 2
```

Порядок сравнения между типами:

```
number < atom < reference < function < port <  
pid < tuple < list < bitstring
```



Объявление переменных

Переменные начинаются с lowercase и могут содержать _

```
x = 10
```

```
user_name = "John"
```

```
_result = 42
```

Иммутабельность - переменные не изменяются, а пересоздаются

```
x = 1
```

```
x = 2
```

 Создается новая переменная

Сопоставление с образцом (Pattern Matching)

```
{a, b} = {1, 2}
```

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

```
iex> x = 1
```

```
1
```

```
iex> x = 2
```

```
2
```

```
iex> {a, b} = {1, 2}
```

```
{1, 2}
```

```
iex> a
```

```
1
```

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

```
iex> head
```

```
1
```

```
iex> tail
```

```
[2, 3]
```



Области видимости

Глобальная область видимости (уровень модуля)

```
global_var = 10
```

```
defmodule MyModule do
```

Модульная область видимости

```
module_var = 20
```

```
def my_function do
```

Локальная область видимости функции

```
local_var = 30
```

```
local_var + module_var
```

```
end
```

```
def another_function do
```



Владение и передача данных

```
original_list = [1, 2, 3]
```

```
new_list = [0 | original_list] [0, 1, 2, 3]
```

Попробуем "изменить" хвост new_list:

```
modified_tail = List.replace_at(new_list, 2, 999)  
меняем третий элемент
```

```
iex> modified_tail
```

```
[0, 1, 999, 3]    НОВЫЙ СПИСОК
```

```
iex> new_list
```

```
[0, 1, 2, 3]      не изменился!
```

```
iex> original_list
```

```
[1, 2, 3]         оригинальный список остался  
нетронутым!
```



ВЫВОДЫ

Итоги:

- Динамическая типизация с сильной проверкой
- Богатая система типов: примитивы + пользовательские
- Иммутабельность всех данных
- Сопоставление с образцом
- Ясные области видимости

Источники

- elixir-lang.org — официальный сайт языка
- Wikipedia — статья Elixir (programming language)

