

# Задачи по Nix езикът

## Универсални конфигурации с Nix

Павел Атанасов    Камен Младенов

10.04.2025

# Преговор

- Разгледахме целия Nix език

# Раздел 1

## Подготвяне на средата

# Подготвяне на средата

- Стандартния начин за изпълнение на Nix код (чийто резултат не е деривация) е

- ➊ Това за изпълнение на даден низ с код

```
nix-instantiate --eval -E '1+1'
```

- ➋ Това за изпълнение на файл с код

```
nix-instantiate --eval ./file.nix
```

## nix repl

- Версия 3 на Nix предоставя един много удобен уред: `nix repl`
- “Read Evaluate Print Loop” е интерактивна среда, в която пишеш израз, той се изпълнява, резултата се принтира, и така докато не излезеш
- Допълнително предоставя “глобални” променливи
- Значително по-удобно е за изпълнение на малки изрази (низове) с код и за интерактивно дебъгване

# Включване на Версия 3 командите

## ❶ Временно:

```
nix --extra-experimental-features "nix-command flakes" ...
```

## ❷ Постоянно - в ~/.config/nix/nix.conf или /etc/nix/nix.conf се добавя:

```
experimental-features = nix-command flakes
```

## Раздел 2

### Задачи

# Задачи

- Може (трябва) да използвате всички функции в `builtins`
- **Не** може да използвате функции от `nixpkgs`



# Задача 1

Реализирайте израз, който смята сумата на числата 1, 2 и 3

**Изход:** 6

## Задача 2

Реализирайте функция, която връща пред-последния елемент от подаден списък. Ако списъка има по-малко от два елемента, върнете `null`.

### Примерен вход/изход

`[ 1 2 3 4 ]`  $\mapsto$  `3`

`[ 1 ]`  $\mapsto$  `null`

## Задача 3

Реализирайте функция, която приема атрибутно множество и връща стойността на първия атрибут. Ако няма атрибути върнете `null`.

**Примерен вход/изход**

```
{ a = "First"; b = "Second" } ↦ "First"  
{ } ↦ null
```

## Задача 4

Реализирайте функция, която по подадено цяло число  $x$  връща сумата на всички числа от 1 до  $x$ . Ако  $x$  е по-малко или равно на 1, тогава връща `null`.

### Примерен вход/изход

5  $\mapsto$  15

1  $\mapsto$  `null`

-5  $\mapsto$  `null`

## Задача 5

Реализирайте *функция*, която приема две цели числа:  $x$  и  $y$ . Връща списък с всички числа от  $x$  до  $y$ .

**Примерен вход/изход**

$1\ 3 \mapsto [ 1\ 2\ 3 ]$

$-6\ -4 \mapsto [ -6\ -5\ -4 ]$

$5\ -3 \mapsto [ 5\ 4\ 3\ 2\ 0\ -1\ -2\ -3 ]$

## Задача 6

Реализирайте функция, която приема списък с низове. Трябва да върне всички низове на четни позиции, които започват с главна буква “В”. Позицията на първия елемент е 0.

**Примерен вход/изход**

```
[ "Breath" "Bob" "Beam" "apple" "british" "british" "orange" ]  
↦ [ "Breath" "Beam" "british" ]
```

## Задача 7

Реализирайте функция, която приема списък с елементи. Всеки елемент на четна позиция е низ - името на даден атрибут. На нечетните са стойностите. Генерирайте атрибутно множество от този списък.

**Примерен вход/изход**

`[ "a" 85 "b" { c = null; } ]  $\mapsto$  { a = 85; b = { c = null }; }`

## Задача 8

Реализирайте функция, която приема списък с атрибутни множества. Във всяко атрибутно множество може да има атрибут “names”. При наличие, този атрибут ще съдържа списък с низове. Върнете всички уникални низове, съдържащи се във “names” на всички атрибутни множества.

### Примерен вход/изход

```
[ { a = 5; } { names = [ "First" "First" "Second" ]; }  
  { names = [ "Second" "Third" ]; k = 10; } ]
```

```
⇒ [ "First" "Second" "Third" ]
```



## Задача 9

Реализирайте функция, която приема атрибутно множество. Всеки негов атрибут също е атрибутно множество, нека да го наречем *програма*. Всяка *програма* съдържа низ име **name** и число версия **version**.

Върнете атрибутно множество с два атрибута: **latest** и **oldest**. И двата са списъка, като първия съдържа уникалните *програми* (спрямо името) с най-новите им версии, докато второто съдържа уникалните *програми* с най-старите. Ако има само една версия, стойността ще е еднаква.

## Примерен вход/изход

```
{ a = { name = "first"; version = 5; };  
  b = { name = "first"; version = 11; };  
  c = { name = "second"; version = 1; }; }
```

⇒

```
{  
  latest = [  
    { name = "first"; version = 11; }  
    { name = "second"; version = 1; }  
  ];  
  oldest = [  
    { name = "first"; version = 5; }  
    { name = "second"; version = 1; }  
  ];  
}
```

## Задача 10

Реализирайте функция, която по подаден списък от атрибути множества връща атрибутно множество, което е **рекурсивна** смесица от всички атрибути множества. Рекурсивно се обновяват, като ако за даден индекс има няколко възможни стойности, то всичките се обединяват в един списък.

## Примерен вход/изход

```
[ { a = 6; b = 10; c.d = [ 8 9 ]; e = "hello"; }  
  { aa = 7; b = 11; c.g = 3.14; e.m = true; } ]
```

$\mapsto$

```
{ a = 6; aa = 7; b = [ 10 11 ];  
  c = { d = [ 8 9 ]; g = 3.14; };  
  e = [ "hello" { m = true; } ]; }
```