

Idris

2014 г.

Idris

- ▶ Haskell-подобный,
- ▶ с зависимыми типами,
- ▶ строгий по-умолчанию,
- ▶ с опциональной проверкой на тотальность,
- ▶ ...

Haskell-подобный

```
data MyList a = Nil | (::) a (MyList a)
```

```
(++) : MyList a → MyList a → MyList a
```

```
[] ++ ys = ys
```

```
(x :: xs) ++ ys = x :: (xs ++ ys)
```

```
instance Functor MyList where
```

```
  map f Nil = Nil
```

```
  map f (x :: xs) = f x :: map f xs
```

Haskell-подобный

instance *Applicative* *MyList* **where**

pure *x* = [*x*]

[] <\$> _ = []

(*f* :: *fs*) <\$> *xs* = *map* *f* *xs* ++ (*fs* <\$> *xs*)

instance *Monad* *MyList* **where**

[] >= _ = []

(*x* :: *xs*) >= *f* = *f* *x* ++ (*xs* >= *f*)

test : *MyList* *Int*

test = *do*

f ← [*id*, (*2)]

x ← [3, 4]

return \$ *f* *x*

С зависимыми типами

data *MyVect* : *Nat* → (*a* : *Type*) → *Type* **where**

Nil : *MyVect* 0 *a*

(::) : *a* → *MyVect* *n* *a* → *MyVect* (*S* *n*) *a*

(++) : *MyVect* *n* *a* → *MyVect* *m* *a* → *MyVect* (*n* + *m*) *a*

[] ++ *ys* = *ys*

(*x* :: *xs*) ++ *ys* = *x* :: (*xs* ++ *ys*)

infix 9 !!

(!!) : *MyVect* *n* *a* → *Fin* *n* → *a*

(*x* :: *xs*) !! *fZ* = *x*

(*x* :: *xs*) !! (*fS* *y*) = *xs* !! *y*

Строгий по-умолчанию

С опциональной проверкой на тотальность

