

# lec 9

2024-11-11

Nat : Type

S : Nat → Nat

O : Nat

Bool : Type

False : Bool

True : Bool

Int : Type

MkInt : Nat → Int

Neg : Int → Int

ou ..... wrapper

MkInt : Nat → Int

ou .....  
MkInt : Nat → Int  
que tal Sign?  
que tal Bool?

ou .....

Z : Int

SP : Int → Int

SN : Int → Int

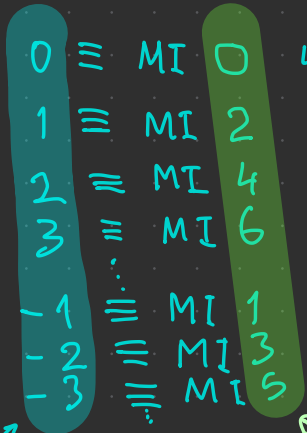
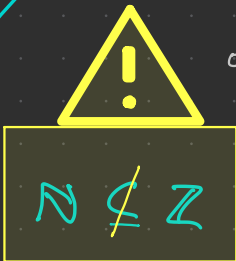
3 ≡ SP (SP (SP Z))

-2 ≡ SN (SN Z)

0 ≡ Z

SN (SP (SP Z))

```
data Sign
  P : Sign
  Z : Sign
  N : Sign
```



Ints

Nats

especificação de inteiros

```
(Z; +, -, 0, 1, Pos) + axiomas
Z_N def= {0_Z, 1_Z, 2_Z, ...} = {0} ∪ Pos
```

# Polimorfismo

$\text{idNat} : \text{Nat} \rightarrow \text{Nat}$

$\text{idNat } n = n$

$\text{idBool} : \text{Bool} \rightarrow \text{Bool}$

$\text{idBool } b = b$

$\text{id} : \alpha \rightarrow \alpha$   
 $\alpha : \text{Type}$

$\text{id } x = x$

isso define uma  
 $\alpha$ -indexada família de funções

("open-ended")

ListNat : Type

$\text{Cons} : \text{Nat} \rightarrow \text{ListNat} \rightarrow \text{ListNat}$

$\text{Nil} : \text{ListNat}$

$\text{List} : \text{Ty} \rightarrow \text{Ty} \quad \alpha : \text{Ty}$

$\text{List } \alpha : \text{Ty}$

data List  $\alpha$

$\text{Nil} : \text{List } \alpha$

$\text{Cons} : \alpha \rightarrow \text{List } \alpha \rightarrow \text{List } \alpha$

$\text{length} : \text{List } \alpha \rightarrow \text{Nat}$

$\text{sum} : \text{List Nat} \rightarrow \text{Nat}$