

# Array specification

ARRAY ( Elem: [Undefined → Elem] )

**sort** Array  
**imports** INTEGER

Arrays are collections of elements of generic type Elem. They have a lower and upper bound (discovered by the operations First and Last). Individual elements are accessed via their numeric index. Create takes the array bounds as parameters and creates the array, initialising its values to Undefined. Assign creates a new array which is the same as its input with the specified element assigned the given value. Eval reveals the value of a specified element. If an attempt is made to access a value outside the bounds of the array, the value is undefined.

Create (Integer, Integer) → Array  
Assign (Array, Integer, Elem) → Array  
First (Array) → Integer  
Last (Array) → Integer  
Eval (Array, Integer) → Elem

First (Create (x, y)) = x  
First (Assign (a, n, v)) = First (a)  
Last (Create (x, y)) = y  
Last (Assign (a, n, v)) = Last (a)  
Eval (Create (x, y), n) = Undefined  
Eval (Assign (a, n, v), m) =  
    if m < First (a) or m > Last (a) then Undefined else  
    if m = n then v else Eval (a,  
m)

# List specification

LIST ( Elem: [Undefined → Elem] )

**sort** List  
**imports** INTEGER

Defines a list where elements are added at the end and removed from the front. The operations are Create, which brings an empty list into existence, Cons, which creates a new list with an added member, Length, which evaluates the list size, Head, which evaluates the front element of the list, and Tail, which creates a list by removing the head from its input list.

Create → List  
Cons (List, Elem) → List  
Tail (List) → List  
Head (List) → Elem  
Length (List) → Integer

Head (Create) = Undefined -- Error to evaluate an empty list  
Head (Cons (L, v)) = if L = Create then v else Head (L)  
Length (Create) = 0  
Length (Cons (L, v)) = Length (L) + 1  
Tail (Create) = Create  
Tail (Cons (L, v)) = if L = Create then Create else Cons (Tail (L), v)

