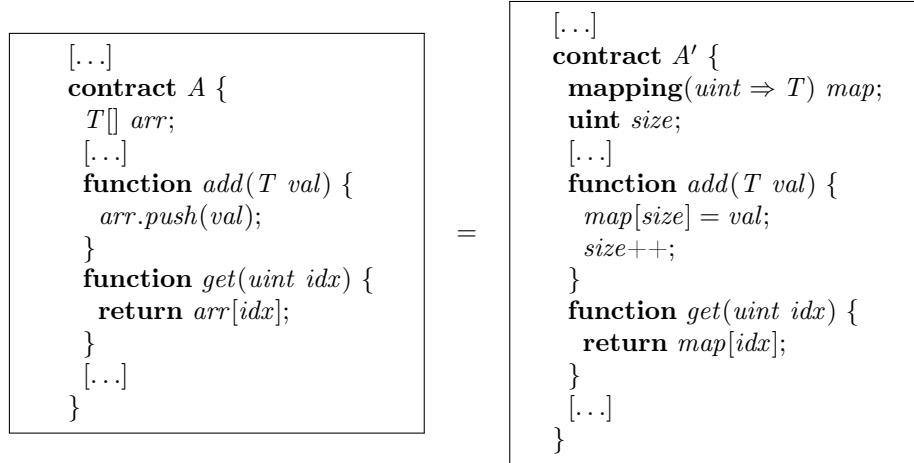


---

**Rule 0.28** *(Use Mappings Instead of Arrays for Data Lists)*

---

**where**

*arr* is a dynamic array of type *T[]* in contract *A*;

*map* is a mapping from **uint** to *T* in contract *A'*;

*size* is a counter tracking the number of elements in the mapping;

*T* is the element type of the array and mapping values;

*val* is a value of type *T* being added;

*idx* is an index used to access elements.

**provided**

The contract does not require iterating over all elements frequently;

Element access is primarily done by index/key rather than sequential iteration;

The mapping provides sufficient functionality for the use case;

A separate *size* counter is maintained to track the number of elements;

Array operations like **push** are replaced with direct mapping assignments and size increments;

Bounds checking uses *size* instead of *arr.length*.

**Invariant:**

Let *s<sub>i</sub>* and *s'<sub>i</sub>* be the initial state of *A* and *A'*, respectively.

Let *s<sub>f</sub>* and *s'<sub>f</sub>* be the state reached by *A* and *A'*, respectively, after *A.f()* and *A'.f()* are executed from *s<sub>i</sub>* and *s'<sub>i</sub>*, respectively.

Then, the coupling invariant is

$$\forall s_i, s'_i . (s_i = s'_i) \rightarrow (s_f = s'_f)$$

---