- The container in which we store key - value pairs, and where the keys are unique and they are in no particular order is the **ADT Map** (or Dictionary)
  - Domain of the ADT Map:

$\mathcal{M} = \{m | \text{m is a map with elements } e =< k, v >, \text{ where } k \in TKey \text{ and } v \in TValue\}$

- init(m)
  - **descr:** creates a new empty map
  - **pre:** true
  - **post:** $m \in \mathcal{M}$, m is an empty map.

- destroy(m)
  - **descr:** destroys a map
  - **pre:** $m \in \mathcal{M}$
  - **post:** m was destroyed

- add(m, k, v)
  - **descr:** add a new key-value pair to the map (the operation can be called *put* as well). If the key is already in the map, the corresponding value will be replaced with the new one. The operation returns the old value, or 0_{TValue} if the key was not in the map yet.
  - **pre:** $m \in \mathcal{M}, k \in TKey, v \in TValue$
  - **post:** $m' \in \mathcal{M}, m' = m \cup < k, v >, add \leftarrow v', v' \in TValue$ where

$$v' \leftarrow \begin{cases} v'', & \text{if } \exists < k, v'' > \in m \\ 0_{TValue}, & \text{otherwise} \end{cases}$$

- remove(m, k)
    - **descr:** removes a pair with a given key from the map. Returns the value associated with the key, or 0_{TValue} if the key is not in the map.
    - **pre:**$m \in \mathcal{M}, k \in TKey$
    - **post:** $remove \leftarrow v$, $v \in TValue$, where

$$v \leftarrow \begin{cases} v', & \text{if } \exists < k, v' > \in m \text{ and } m' \in \mathcal{M}, \\ & m' = m \backslash < k, v' > \\ 0_{TValue}, & \text{otherwise} \end{cases}$$

- search(m, k)
    - **descr:** searches for the value associated with a given key in the map
    - **pre:** $m \in \mathcal{M}, k \in TKey$
    - **post:** $search \leftarrow v$, $v \in TValue$, where

$$v \leftarrow \begin{cases} v', & \text{if } \exists < k, v' > \in m \\ 0_{TValue}, & \text{otherwise} \end{cases}$$

- iterator(m, it)
    - **descr:** returns an iterator for a map
    - **pre:** $m \in \mathcal{M}$
    - **post:** $it \in \mathcal{I}$, it is an iterator over m.

- **Obs:** The iterator for the map is similar to the iterator for other ADTs, but the *getCurrent* operation returns a <key, value> pair.

- size(m)
    - **descr:** returns the number of pairs from the map
    - **pre:**$m \in \mathcal{M}$
    - **post:** size $\leftarrow$ the number of pairs from m

- isEmpty(m)
  - **descr:** verifies if the map is empty
  - **pre:** $m \in \mathcal{M}$
  - **post:** $isEmpty \leftarrow \begin{cases} true, & \text{if m contains no pairs} \\ false, & \text{otherwise} \end{cases}$
- keys(m, s)
  - **descr:** returns the set of keys from the map
  - **pre:**$m \in \mathcal{M}$
  - **post:**$s \in \mathcal{S}$, s is the set of all keys from m
- values(m, b)
  - **descr:** returns a bag with all the values from the map
  - **pre:** $m \in \mathcal{M}$
  - **post:**$b \in \mathcal{B}$, b is the bag of all values from m
- pairs(m, s)
  - **descr:** returns the set of pairs from the map
  - **pre:**$m \in \mathcal{M}$
  - **post:**$s \in \mathcal{S}$, s is the set of all pairs from m
- We can have a Map where we can define an order (a relation) on the set of possible keys
- The only change in the interface is for the *init* operation that will receive the *relation* as parameter.
- For a sorted map, the iterator has to iterate through the pairs in the order given by the *relation*, and the operations *keys* and *pairs* return SortedSets.