CSE-4301
Object Oriented Programming
2022-2023

Week-14

The Standard Template Library

-Faisal Hussain
Department Of Computer Science And Engineering, IUT

Contents

Introduction to STL

- Three important entities :
  - Container
  - Algorithm
  - Iterator

Container

- A way that stored data is organized in memory
  - Stacks, link list, array
- STL containers are created with template classes
  - It can be customized to handle different data type
- No need to specify the size of STL containers. The containers themselves. take care of all memory allocation.

Algorithm

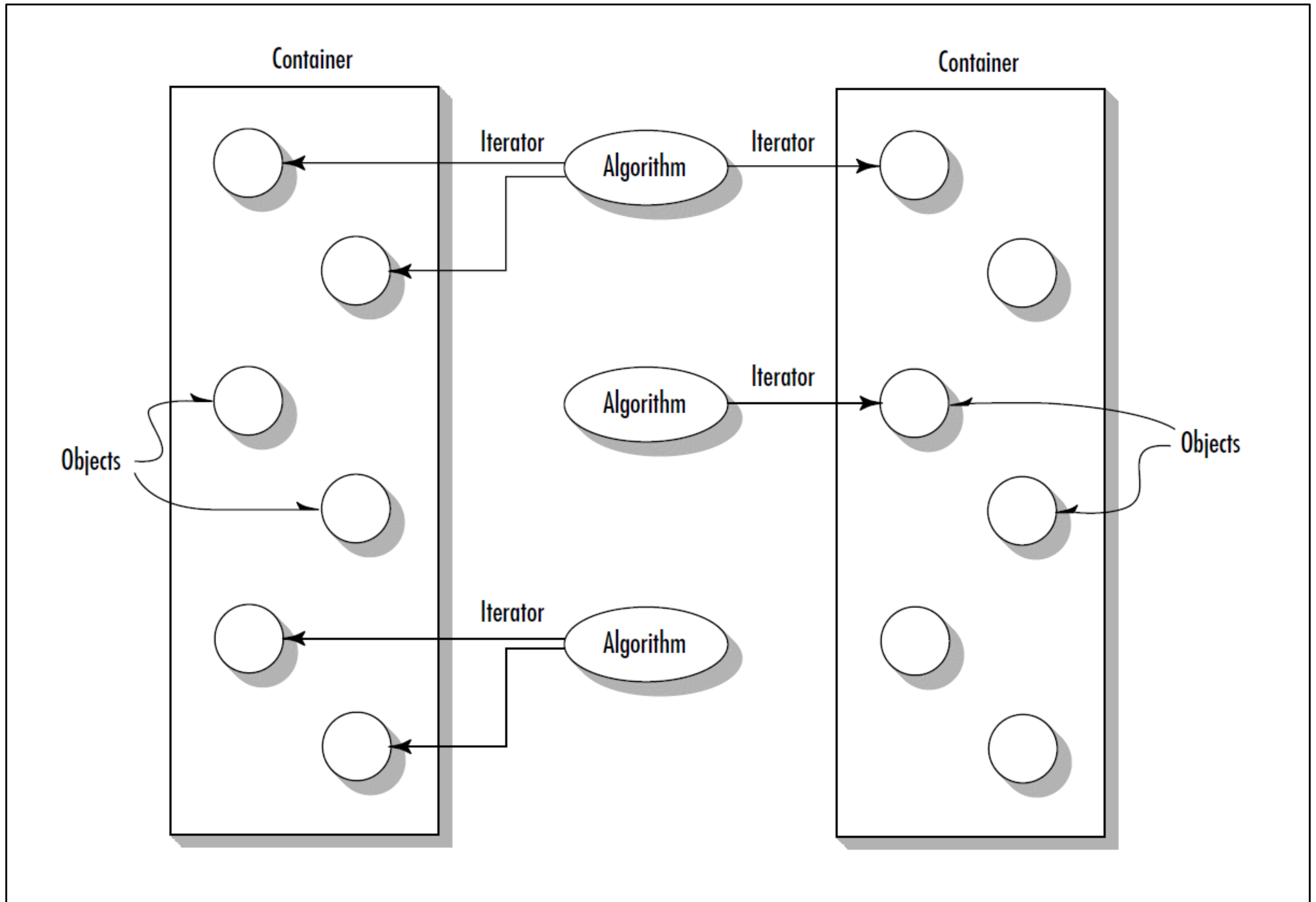
- Procedures that are applied to **containers** to process their data in various ways.
  - sort, copy, search, and merge data

- They are not member of container classes rather independent function

- STL algorithms are general can work on STL containers, C++ standard container array or user defined container

Iterator

- Iterators are a generalization of the concept of pointers: they point to elements in a container.
- Iterators are a key part of the STL because they connect algorithms with containers



Algorithms use iterators to act on objects in containers

Container

- Two main categories:
  - Sequence
  - Associative

- Sequence: *vector*, *list*, and *deque*
- Associative: *set*, *multiset*, *map*, and *multimap*.

Sequence Container

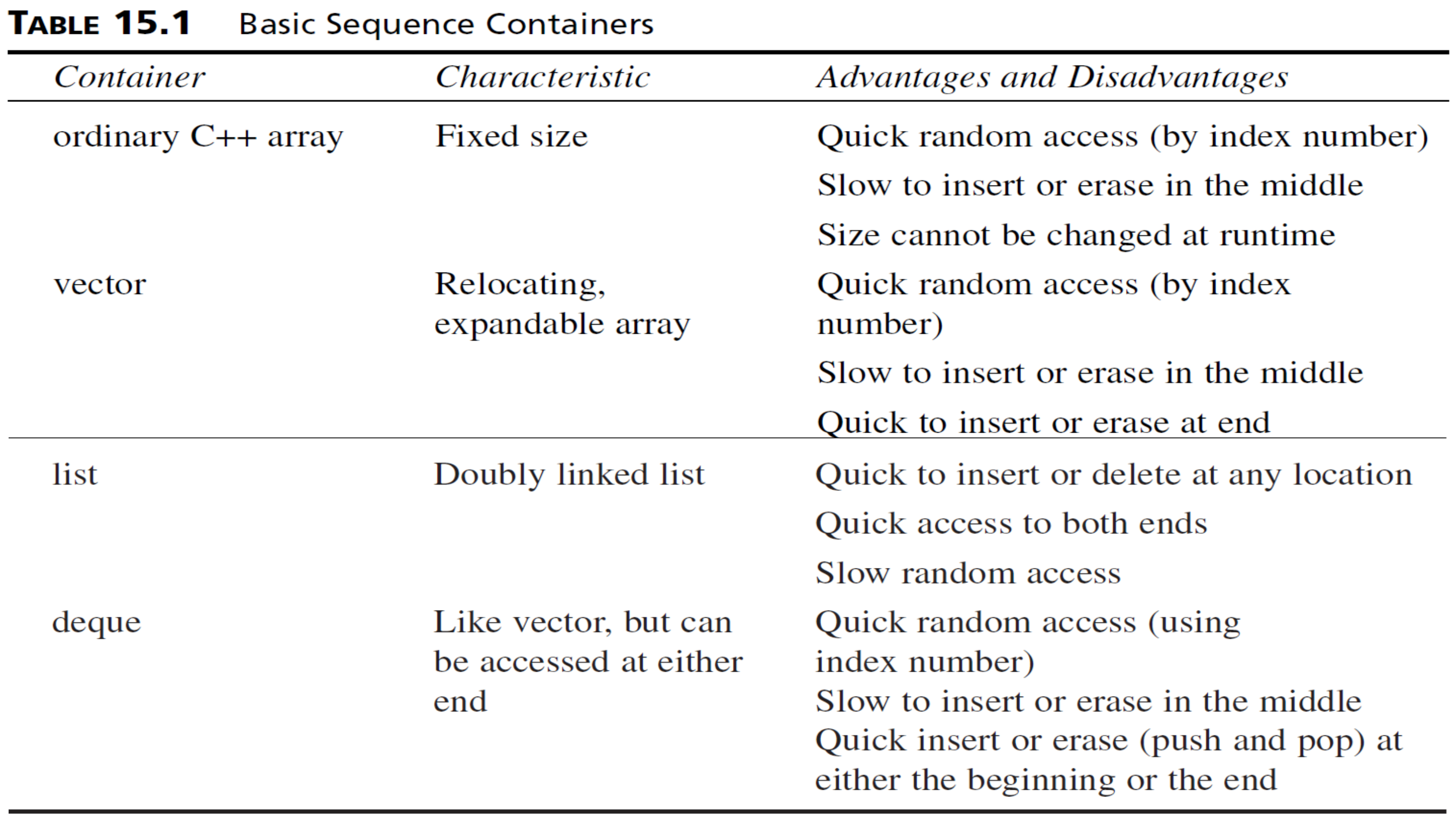


TABLE 15.1 Basic Sequence Containers

Container	*Characteristic*	*Advantages and Disadvantages*
ordinary C++ array	Fixed size	Quick random access (by index number) Slow to insert or erase in the middle Size cannot be changed at runtime
vector	Relocating, expandable array	Quick random access (by index number) Slow to insert or erase in the middle Quick to insert or erase at end
list	Doubly linked list	Quick to insert or delete at any location Quick access to both ends Slow random access
deque	Like vector, but can be accessed at either end	Quick random access (using index number) Slow to insert or erase in the middle Quick insert or erase (push and pop) at either the beginning or the end

- vector<int> aVect; //create a vector of ints
- list<airtime> departure_list; //create a list of airtimes

Associative Container

- An associative container is not sequential; instead it uses *keys* to access data.

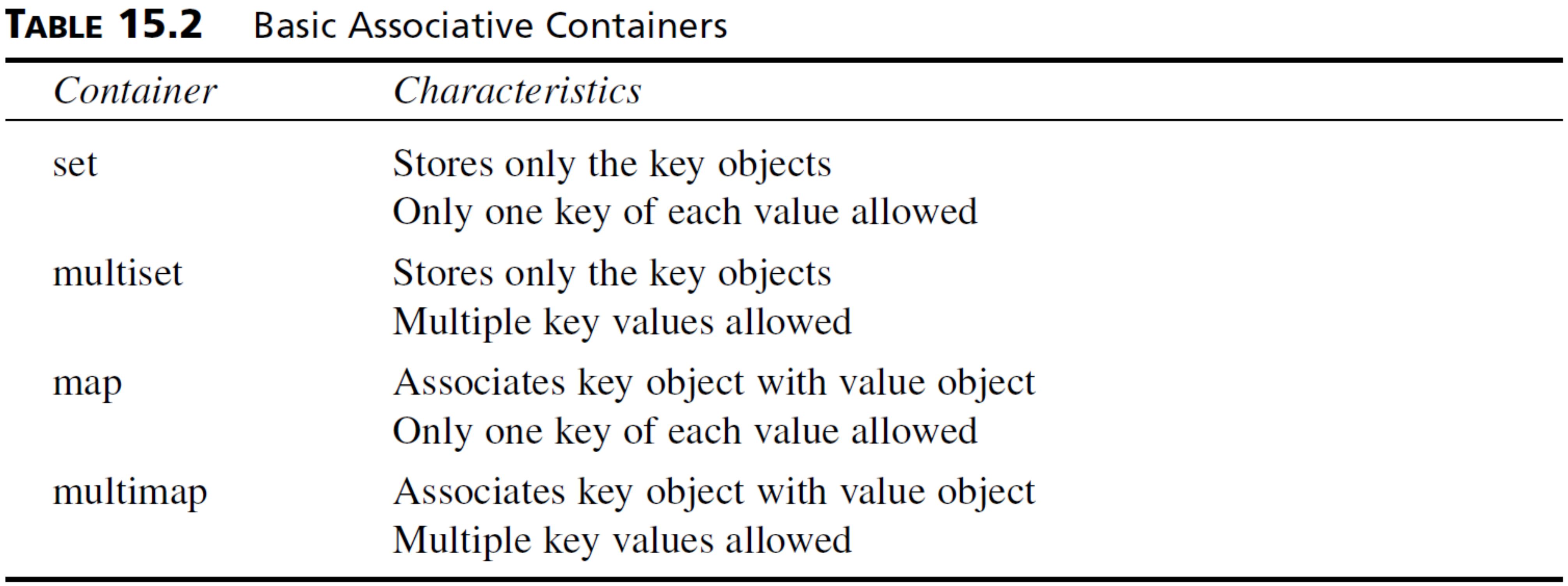


TABLE 15.2 Basic Associative Containers

Container	*Characteristics*
set	Stores only the key objects Only one key of each value allowed
multiset	Stores only the key objects Multiple key values allowed
map	Associates key object with value object Only one key of each value allowed
multimap	Associates key object with value object Multiple key values allowed

- set<int> intSet; //create a set of ints
- multiset<employee> machinists; //create a multiset of employees

Member function common all containers

TABLE 15.3 Some Member Functions Common to All Containers

Name	*Purpose*
`size()`	Returns the number of items in the container
`empty()`	Returns true if container is empty
`max_size()`	Returns size of the largest possible container
`begin()`	Returns an iterator to the start of the container, for iterating forwards through the container
`end()`	Returns an iterator to the past-the-end location in the container, used to end forward iteration
`rbegin()`	Returns a reverse iterator to the end of the container, for iterating backward through the container
`rend()`	Returns a reverse iterator to the beginning of the container; used to end backward iteration

Algorithms

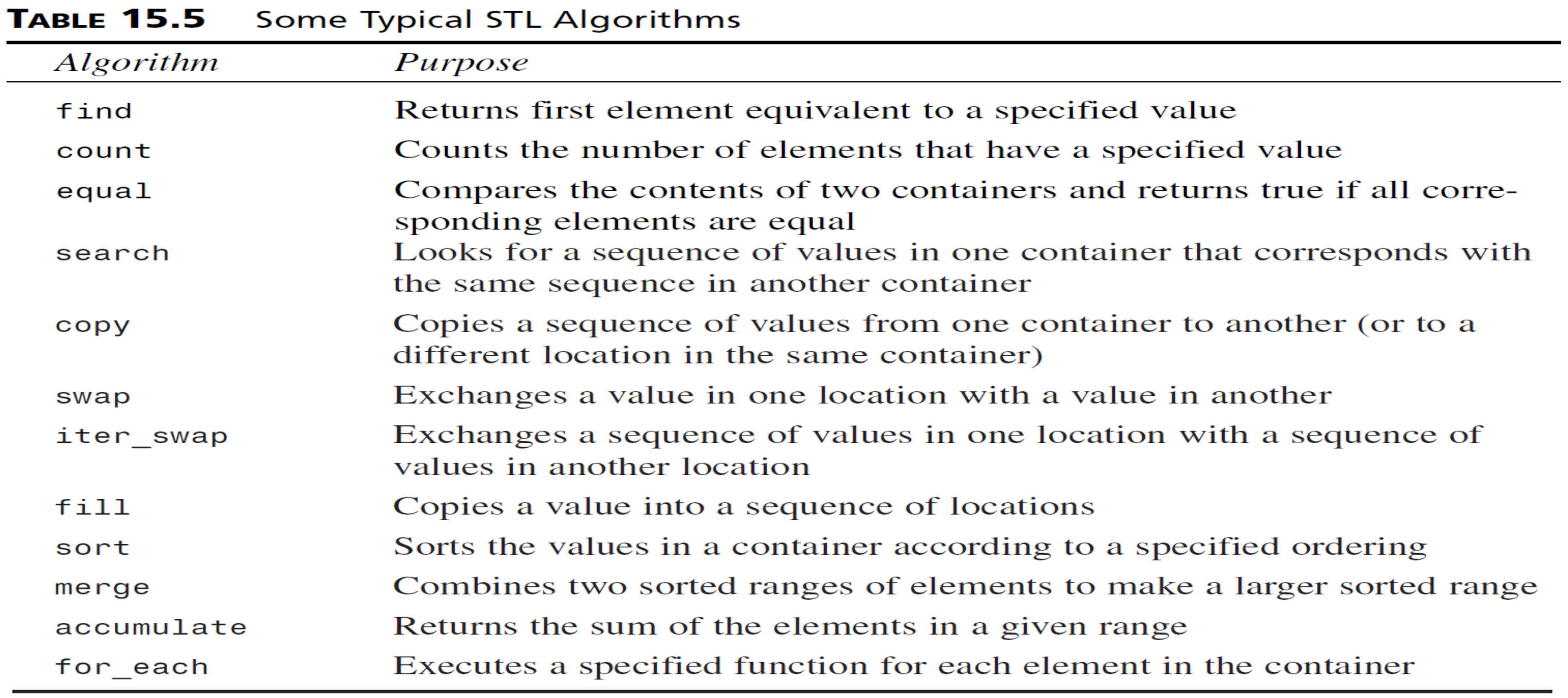


TABLE 15.5 Some Typical STL Algorithms

Algorithm	*Purpose*
`find`	Returns first element equivalent to a specified value
`count`	Counts the number of elements that have a specified value
`equal`	Compares the contents of two containers and returns true if all corresponding elements are equal
`search`	Looks for a sequence of values in one container that corresponds with the same sequence in another container
`copy`	Copies a sequence of values from one container to another (or to a different location in the same container)
`swap`	Exchanges a value in one location with a value in another
`iter_swap`	Exchanges a sequence of values in one location with a sequence of values in another location
`fill`	Copies a value into a sequence of locations
`sort`	Sorts the values in a container according to a specified ordering
`merge`	Combines two sorted ranges of elements to make a larger sorted range
`accumulate`	Returns the sum of the elements in a given range
`for_each`	Executes a specified function for each element in the container

- int arr[8] = {42, 31, 7, 80, 2, 26, 19, 75};
- sort(arr, arr+8);

Iterator

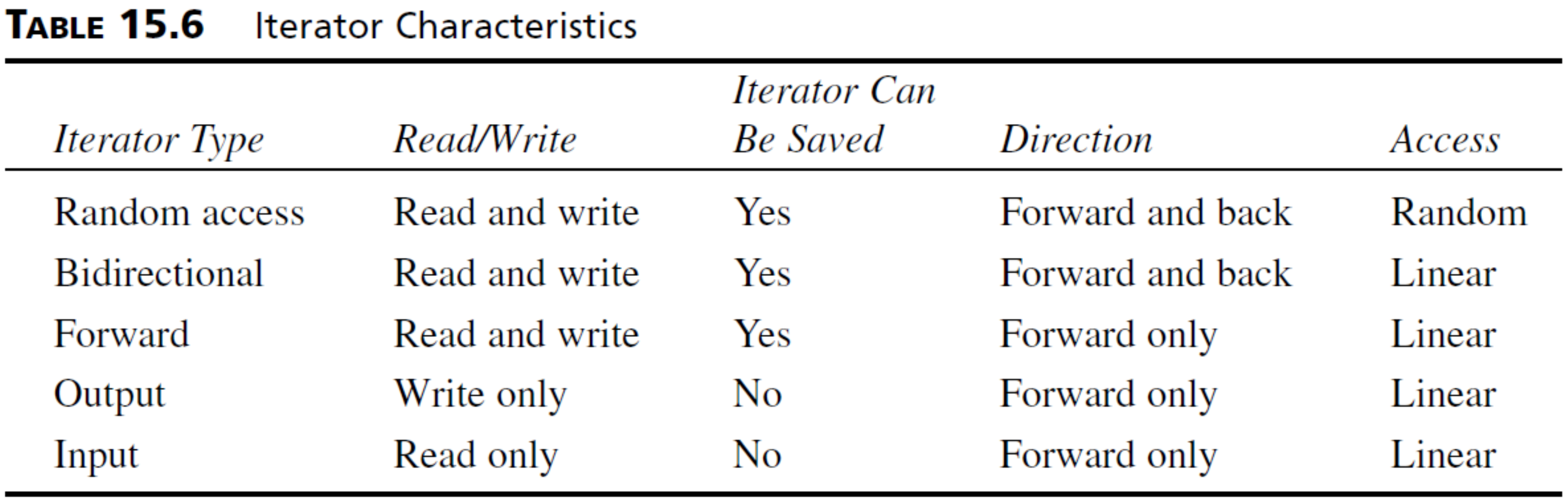


TABLE 15.6 Iterator Characteristics

Iterator Type	*Read/Write*	*Iterator Can Be Saved*	*Direction*	*Access*
Random access	Read and write	Yes	Forward and back	Random
Bidirectional	Read and write	Yes	Forward and back	Linear
Forward	Read and write	Yes	Forward only	Linear
Output	Write only	No	Forward only	Linear
Input	Read only	No	Forward only	Linear