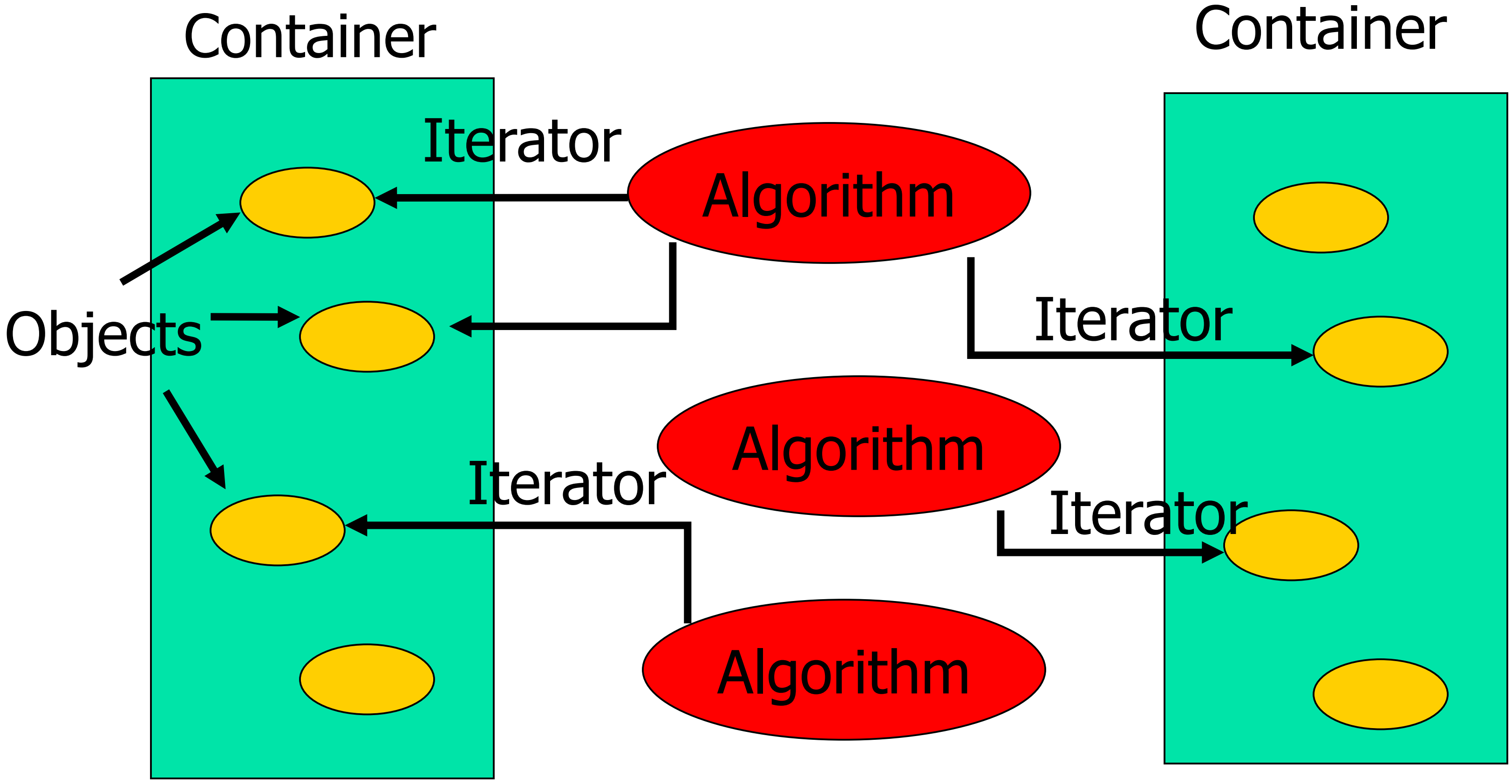
Standard Template Library

- The standard template library (STL) contains
  - **CONTAINERS**
  - **ALGORITHMS**
  - **ITERATORS**
- A ***container*** is a way that stored data is organized in memory, for example an array of elements.
- ***Algorithms*** in the STL are procedures that are applied to containers to process their data, for example search for an element in an array, or sort an array.
- ***Iterators*** are a generalization of the concept of pointers, they point to elements in a container, for example you can increment an iterator to point to the next element in an array

Containers, Iterators, Algorithms

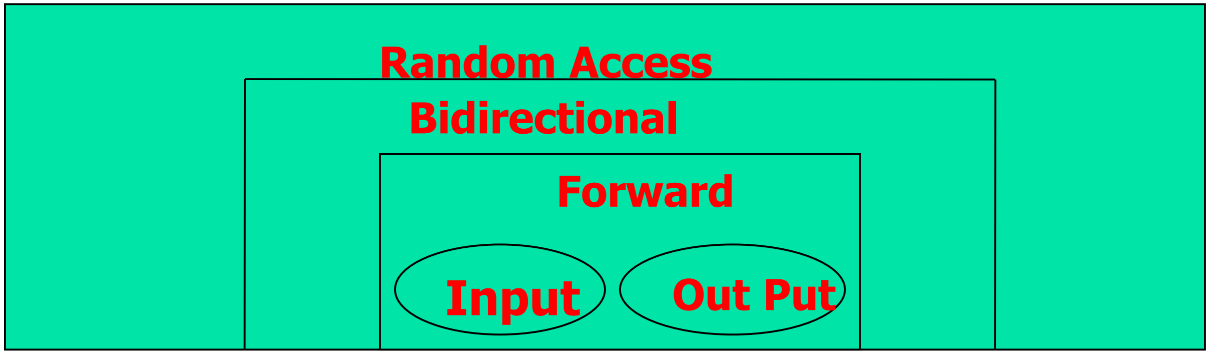
Algorithms use iterators to interact with objects stored in containers



Iterator :

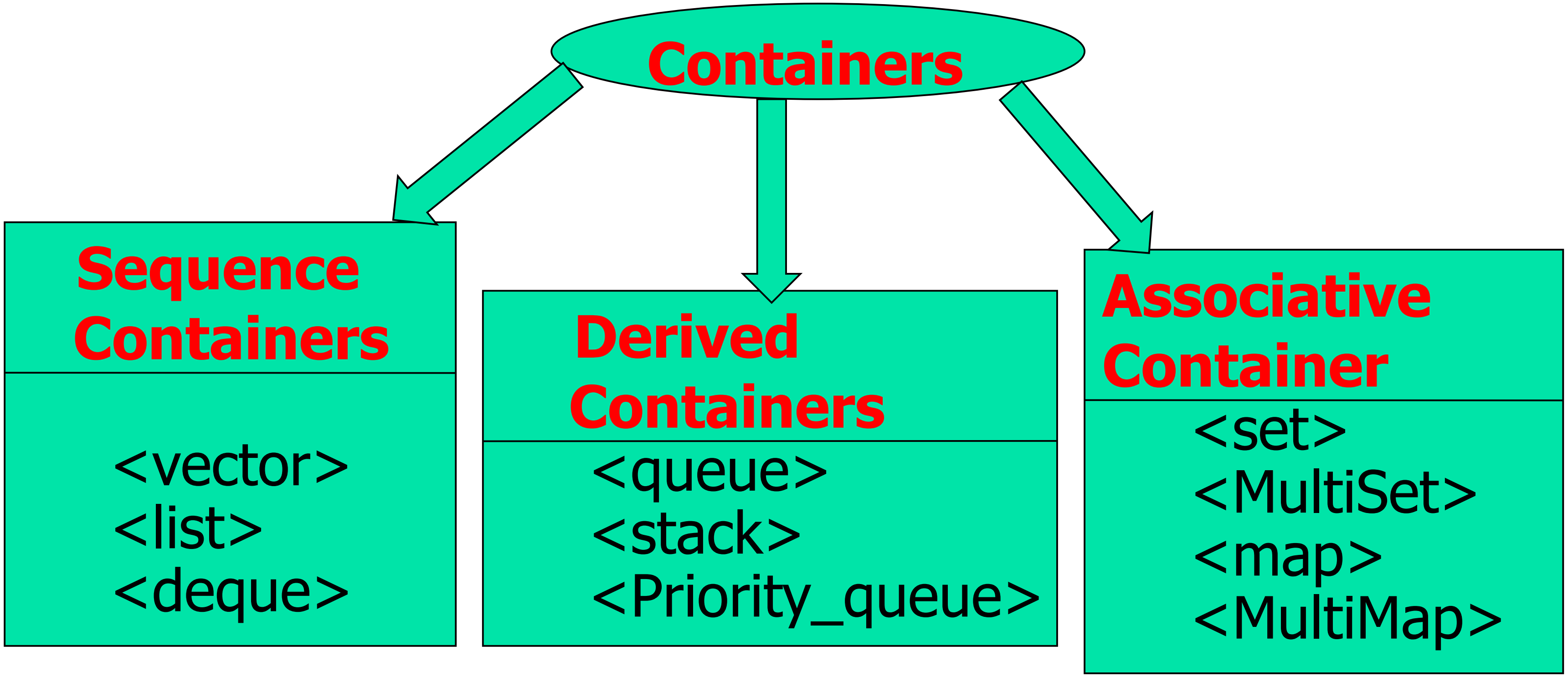
- There are Five Type Of Iterator

ITERATOR	ACCESS METHOD	DIRECTION	I/O CAPABILITY
Input	Liner	Forward Only	Read Only
Output	Liner	Forward Only	Write Only
Forward	Liner	Forward Only	Read/Write
Bidirectional	Liner	Forward /Backward	Read/Write
Random	Random	Forward /Backward	Read/Write



Containers

- A container is a way to store data, either built-in data types like int and float, or class objects
- The STL define ten container which are grouped into three categories



Containers

Sequence Containers

<vector>
<list>
<deque>

Derived Containers

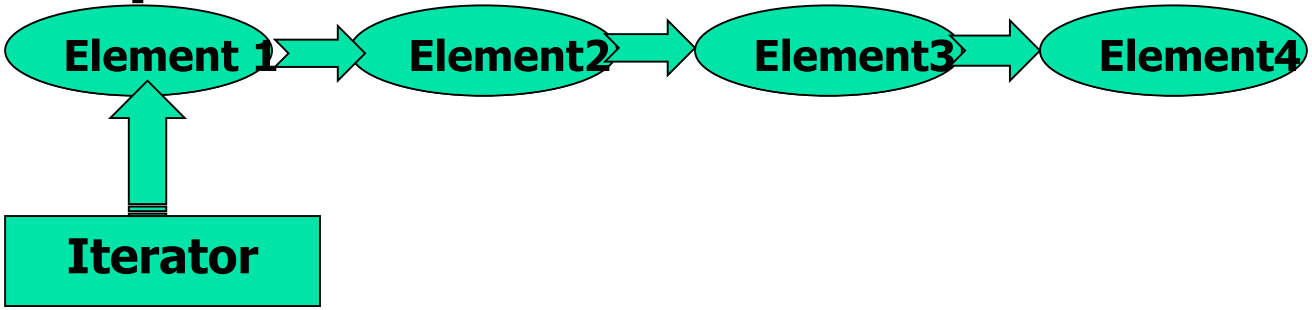
<queue>
<stack>
<Priority_queue>

Associative Container

<set>
<MultiSet>
<map>
<MultiMap>

Sequence Containers

- Sequence containers store elements in a liner sequence. Like line.



Container	Random access	Insertion and Deletion in the Middle	Insertion and Deletion at the ends
vector	Fast	Slow	Fast at back
list	Slow	Fast	Fast at Front
deque	Fast	Slow	Fast at both the Ends

Associative Containers

- An associative container is non-sequential but uses a *key* to access elements. The keys, typically a number or a string, are used by the container to arrange the stored elements in a specific order,for example in a dictionary the entries are ordered alphabetically.
- The main advantage of associative containers is the speed of searching (binary search like in a dictionary)
- Searching is done using a *key* which is usually a single value like a number or string
- The STL contains two basic associative containers
  - sets and multisets
  - maps and multimaps

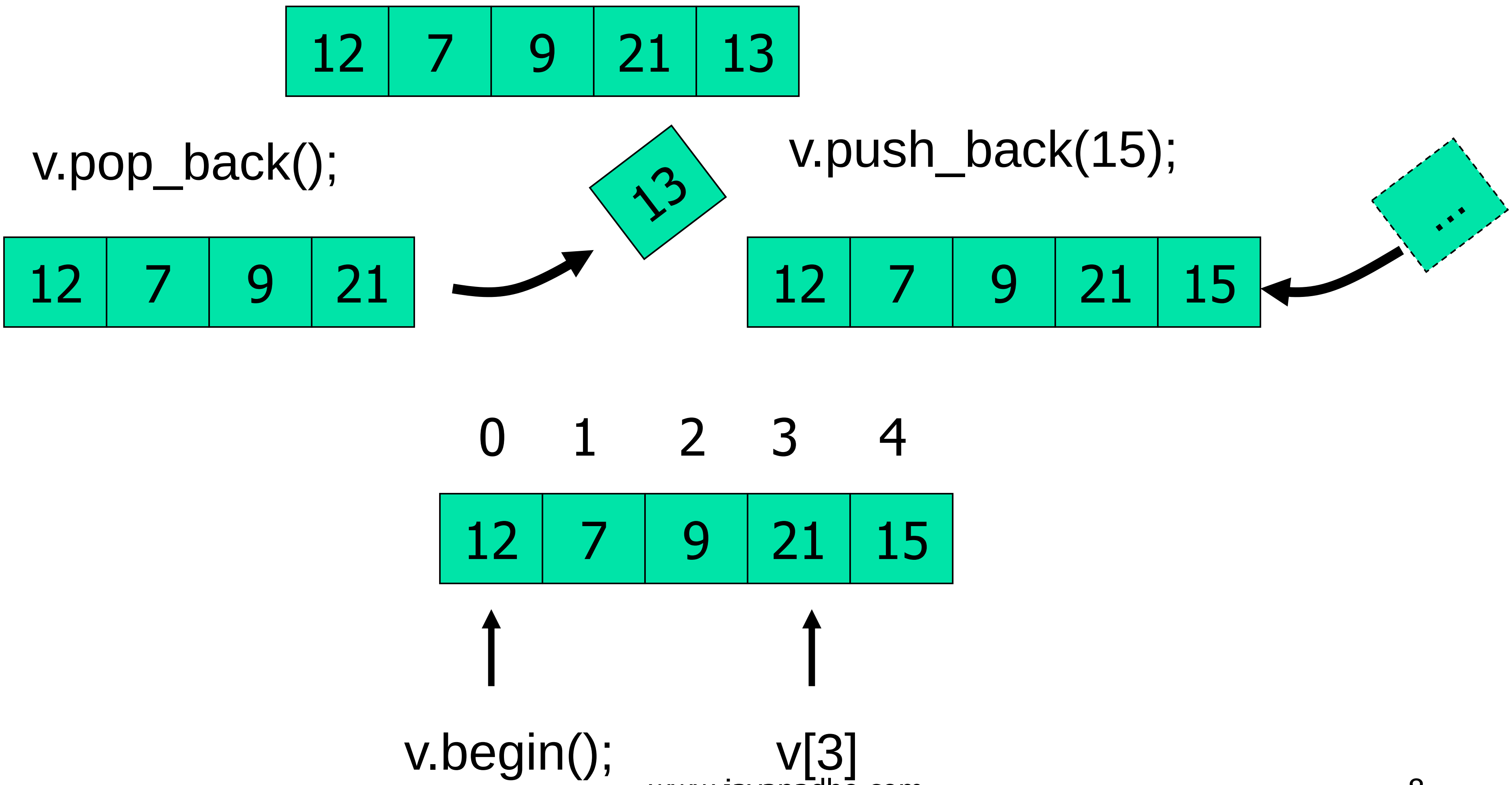
Derived Container

- Deriver container do not support iterators and therefore We can’t use them for data manipulation .they support two member function pop() and push() for implementing deleting and inserting operation

- Stack
- Queue
- Priority Queue

Vector Container

```
int array[5] = {12, 7, 9, 21, 13 };
vector<int> v(array,array+5);
```



v.pop_back();

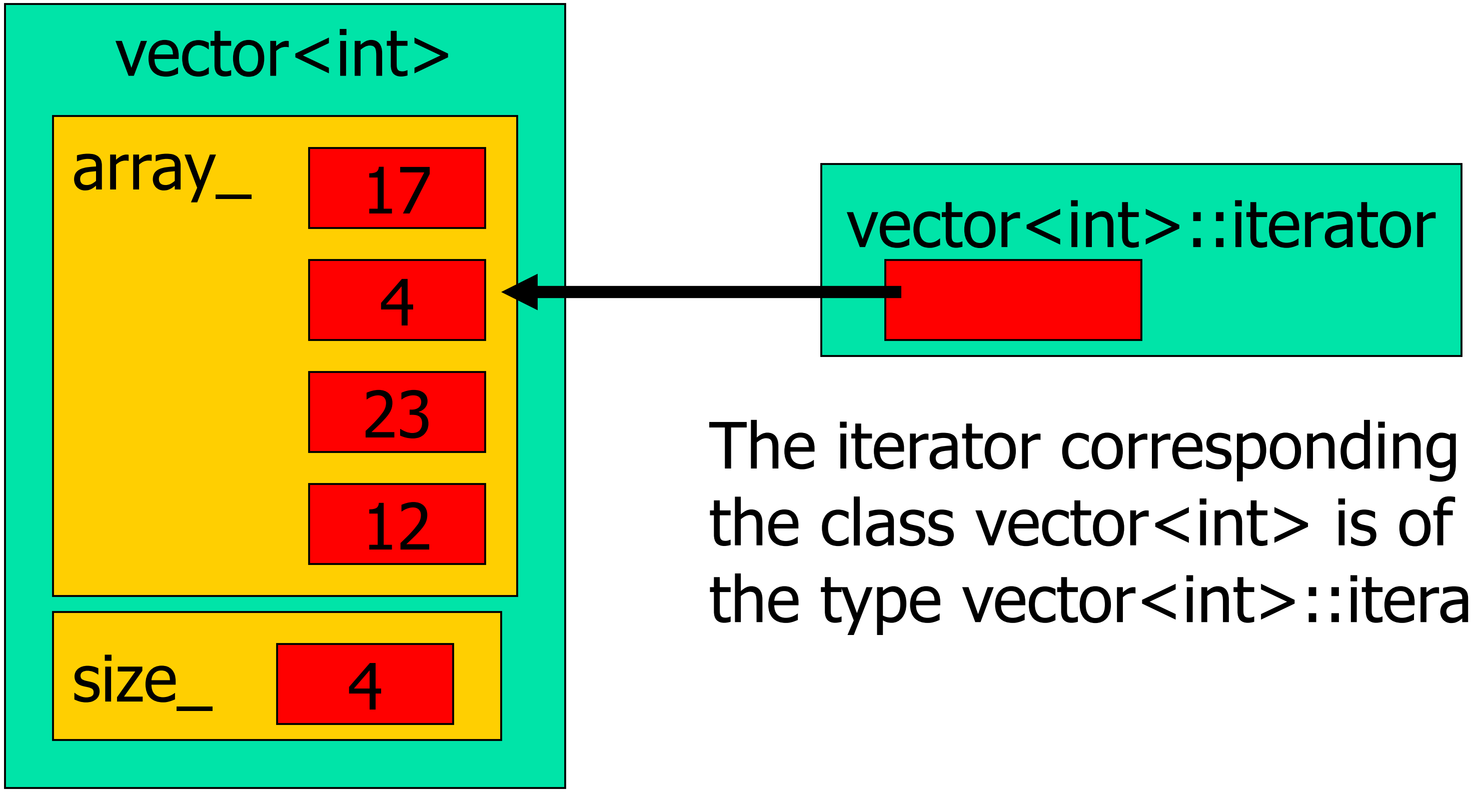
v.push_back(15);

v.begin();

v[3]

Iterators

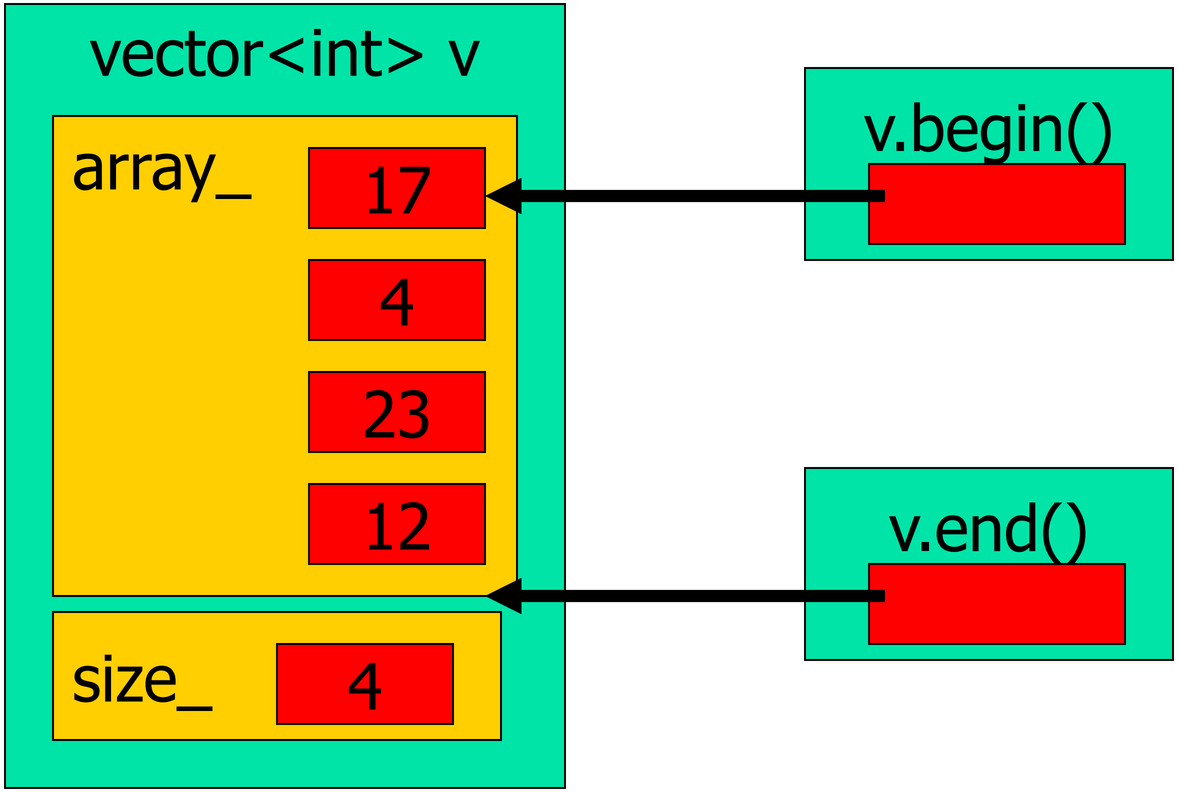
- Iterators are pointer-like entities that are used to access individual elements in a container.
- Often they are used to move sequentially from element to element, a process called *iterating* through a container.



The iterator corresponding to the class vector<int> is of the type vector<int>::iterator

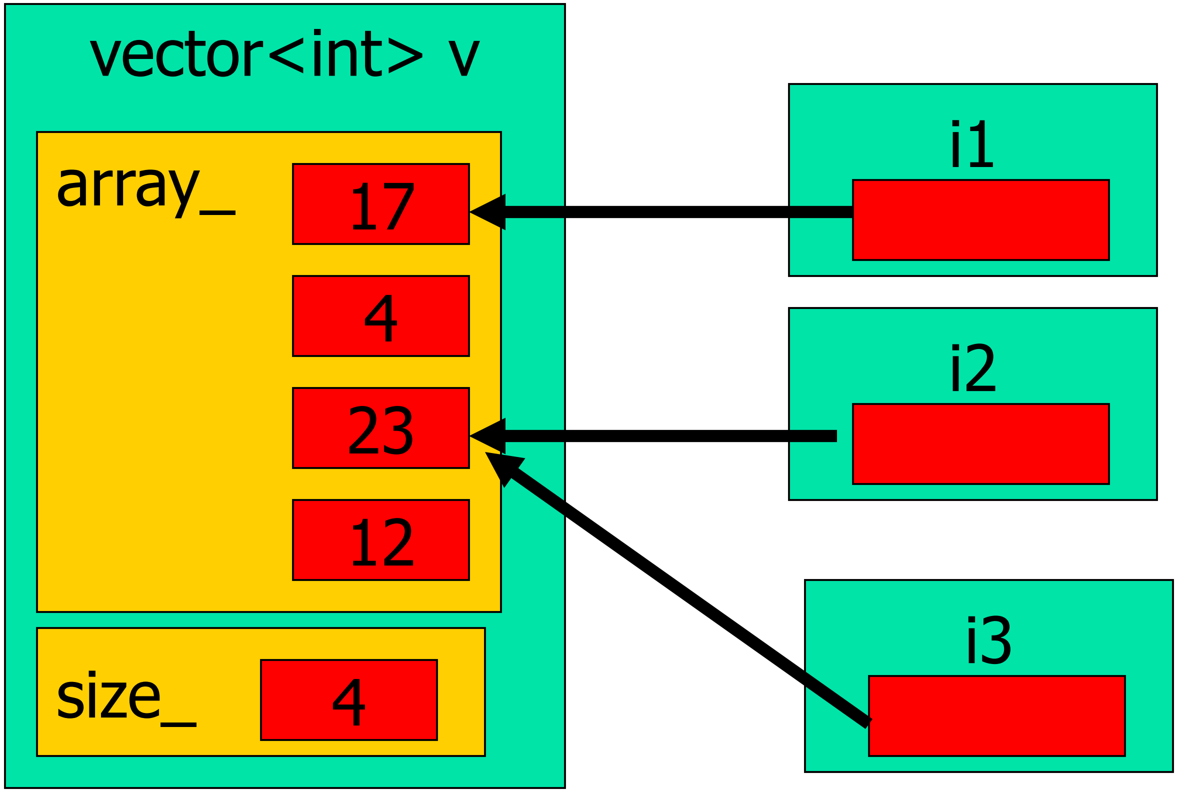
Iterators

- The member functions begin() and end() return an iterator to the first and past the last element of a container



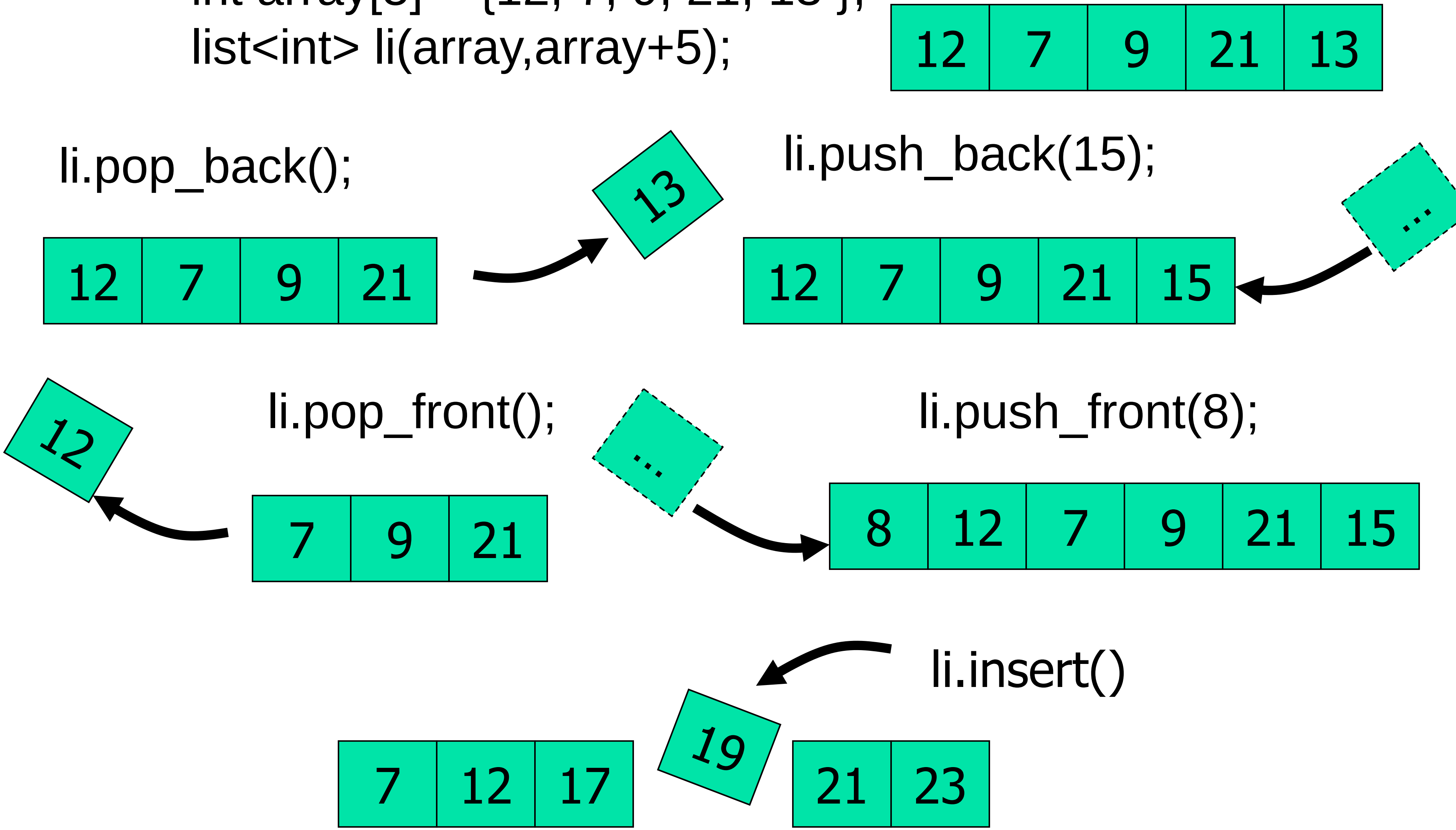
Iterators

- One can have multiple iterators pointing to different or identical elements in the container



List Container

```
int array[5] = {12, 7, 9, 21, 13 };
list<int> li(array,array+5);
```



li.pop_back();

li.push_back(15);

li.pop_front();

li.push_front(8);

li.insert()

Description Of All Container

Container	Descriptor	Header File	Iterator
vector	A dynamic array allow insertion and deletion or back permits direct access to any element.	vector	Random Access
list	A bidirectional liner list allow insertion and deletion any where.	list	Bidirectional
deque	A double ended queue allow insertion and deletion at both the ends permits direct access to any element.	deque	Random access
set	Associative container for storing unique set. All ours rapid lookup (no duplicate allowed)	set	Bidirectional
multiset	An associative container for storing non unique sets(duplicates allowed)	set	Bidirectional

Container	Descriptor	Header File	Iterator
map	An associative container for storing unique key/value pair each key associated with only one value .	map	Bidirectional
multimap	An associative container for storing unique key/value pair in which one key may associated with more than one value.	map	Bidirectional
stack	a standard stack last in first out	stack	Not Iterator
queue	a standard stack first in first out	queue	Not Iterator
priority queue	A priority queue the first element out first element the highest priority element	queue	Not Iterator

Vector program

```
#include<iostream>
#include<vector>
using namespace std;
void display(vector<int> &v)
{   for(int i=0;i<v.size();i++)
    {          cout<<v[i]<<" ";
    }
    cout<<"\n";
}
int main()
{  vector<int> v; // create a vector of type int
   cout<<"Initial size ="<<v.size()<<"\n";
   int x;
   cout<<"Enter five integre values:";
   for(int i=0;i<5;i++) //putting values into the vector
   {     cin>>x;
       v.push_back(x);
    }
   cout<<"Size after adding 5 values:";
   cout<<v.size()<<"\n";
  cout<<"Current Content:\n";
   display(v); // Display the contents
```

```
 // Insert elements
   vector<int>:: iterator itr=v.begin(); //iterator
   itr = itr + 3;
   v.insert(itr,1,9);

   // Display the contents

   cout<<"\n Content after inserting: \n";
   display(v);

   // Removing 4th and 5th elements

   v.erase(v.begin()+3,v.begin()+5);

   // Display the contents
   cout<<"\n Content after deletion:\n";
   display(v);
   cout<<"END \n";


   getch();
   return 0;


}
```