Abstract Data Types

- mathematical abstractions of data types
- An ADT specifies
  - A set of objects
  - A set of operations on the data or subsets

> *Does not specify how the operations should be implemented*

Lists

- $A_0, A_1, A_2, \ldots A_{n-1}$
- The size of the list is N

Iterators

- Need a way to navigate through the items in a container
- A doubly linked list would need a different form than a simple linked list

```
// iterates through the array
for(int i=0; i != v.size(); i++){
        cout << v[i] << endl;
}
```

- A generalized type that helps in navigating any container
  - A way to initialize front and back
  - A way to move to the next
  - A way to detect the end

Getting an Iterator

- tells you the location of the objects
- can be written as

```
for(vector<int>::iterator itr=v.begin(); itr = v.end(); itr++)
        cout << *itr << endl; // must dereference

// use auto as well, C++11
auto itr = v.begin();
```

Adding / Removing

```
iterator insert(iteratorPos...)
iterator remove(iteratorPos...)

// Removing every other element from a list
auto itr = lst.begin();
while( itr != lst.end()){
        itr = lst.erase(itr);
        if(itr != lst.end())
                itr++;
}

// Before c++11
typename Container::iterator itr = lst.begin();
```

Vector in C++ STL

```
int size() // num of elements
void clear() // removes all elements
bool empty() // t or f if empty
void push_back() // put in back of vector
void pop_back() // remove from vector {size--}

// Operators
Object& operator[](index) // return obj index
Object& at (int index) // object at location
int capacity() // internal capacity
void reserve() // set new capacity
void resize() // change the size of a vector (need to copy)
```

Vector Class Template

- Can be copied
- The memory it uses automatically reclaimed
- Maintains primitive array