ITERATORS

ACKNOWLEDGEMENT: THE SLIDES ARE PREPARED FROM SLIDES PROVIDED BY NANCY M. AMATO AND JORY DENNY
A general purpose library of data structures (containers) and algorithms communicating through iterators.

Plug each piece together and use in different applications.

```cpp
#include <vector>
#include <algorithm>
int main(){
  int myArray[4]={2, 8, 5, 3};
  std::vector<int> v(myArray, myArray+4);
  std::sort(v.begin(), v.end());
}
```
CONTAINERS

- Container is an object storing other objects which can be accessed by iterators
- Let $X$ be a container, then some important associated types:
  - $X$::value_type – type of data stored in the container
  - $X$::iterator – type of iterator for the container
  - $X$::const_iterator
  - ...

...
CONTAINERS

- Valid expressions
  - \( a.\text{begin}(\quad) \) – beginning of container
  - \( a.\text{end}(\quad) \) – one past the last element of container
  - \( a.\text{size}(\quad) \) – size of container
  - \( a.\text{empty}(\quad) \) – does the container have anything in it?
Iterators are used to point to other objects.
Iterators play an interface between data structures and algorithms
Main operations
- advance(), next(), operator ++
- Dereference – operator *, operator → ()
ITERATORS

- Forward Iterator – Only allows traversal from begin to end
- Bidirectional Iterator – Allows traversal in both directions (reversed order)
- Random Access Iterator – Arbitrarily access elements in any order
- etc.

- Algorithms are defined for certain types of iterators, i.e., sort is only defined for random access iterating
EXAMPLE

- Create a vector containing integer values
  
  ```cpp
  int myArray[4]={2, 8, 5, 3};
  std::vector<int> v(myArray, myArray+4);
  ```

- Iterate over the vector and output each integer
  ```cpp
  for(vector<int>::iterator it = v.begin(); i != v.end(); ++i)
      cout<<*(it);
  ```

- Iterate over the vector using random access iterators to output each value
  ```cpp
  for(size_t i = 0; i < v.size(); ++i)
      cout<<*(v.begin() + i);
  ```

  99 little bugs in the code.
  99 little bugs.
  Take one down, patch it around.
  127 little bugs in the code...

  **Test your code !!!**  (Pic Courtesy: Michael McKenna)