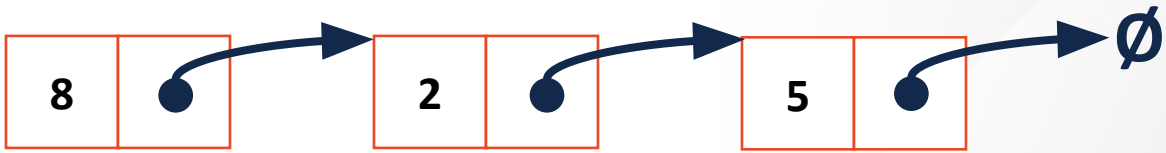
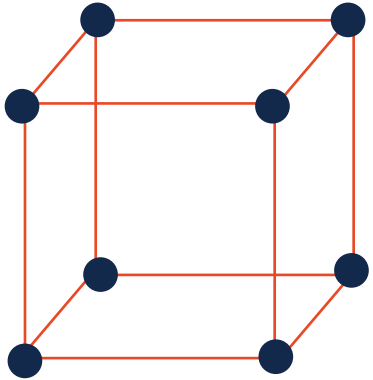
Iterators

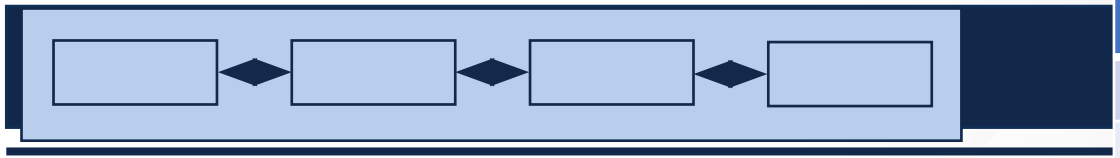
Learning Objectives

1. Describe what an iterator does
2. Enumerate the functions necessary to implement custom iterators
3. Differentiate between pre and post increment operators

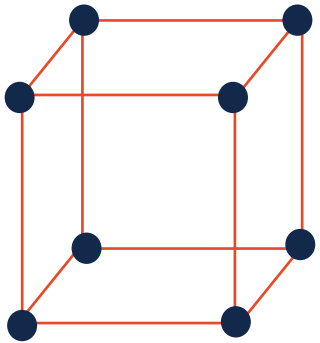
What does an iterator do?





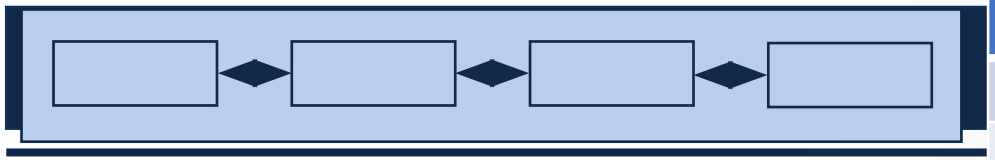


Cur. Location	Cur. Data	Next
index		
ListNode *		
(x, y, z)		

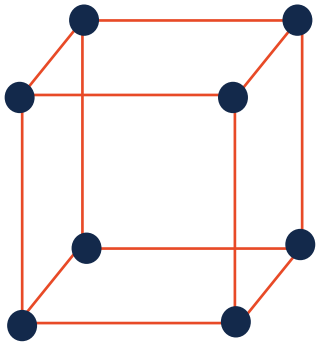


Iterators help answer 2 questions

1. What data is here?
2. What order do I traverse my data structure?



Cur. Location	Cur. Data	Next
index	data[index]	++index
ListNode *	cur->data	cur->next
(x, y, z)	??	??



Iterators help answer 2 questions

1. What data is here?
2. What order do I traverse my data structure?

```cpp
#include <iostream>
#include <vector>

int main() {
    std::vector<int> numbers = {10, 20, 30, 40, 50};
    for (size_t i = 0; i < numbers.size(); ++i) {
        std::cout << numbers[i] << std::endl;
    }
    return 0;
}
```

```cpp
#include <iostream>
#include <vector>

int main() {
    std::vector<int> numbers = {10, 20, 30, 40, 50};
    std::vector<int>::iterator it;
    for (it = numbers.begin(); it != numbers.end(); ++it) {
        std::cout << *it << std::endl;
    }
    return 0;
}
```

Container Functions

begin()

Container Functions

begin()

end()

- One past the end
- Item does not exist

Iterator Functions

Must have the base class std::iterator

Function	Purpose
operator *()	Get data
operator ++()	Go to the next value
operator !=(const Iterator& rhs)	Not Equal

Iterator Functions

Must have the base class std::iterator

Other Functions
->
--
Backwards iterator

Pre vs. Post Increment Operators

```
 1 #include <iostream>
 2 #include <vector>
 3
 4 int main() {
 5     std::vector<int> numbers = {10, 20, 30, 40, 50};
 6     std::vector<int>::iterator it = numbers.begin();
 7     std::cout << *it << std::endl;
 8     std::cout << *(it++) << std::endl;
 9     std::cout << *it << std::endl;
10     std::cout << *(++it) << std::endl;
11     std::cout << *it << std::endl;
12 }
```

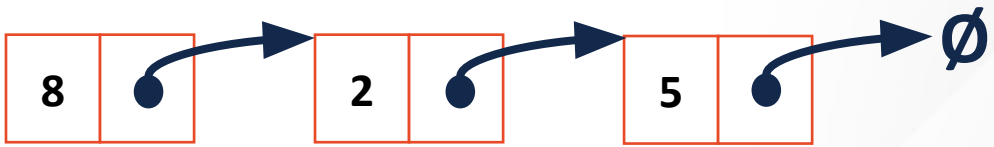
Pre vs. Post Increment Operators

```
 1 #include <iostream>
 2 #include <vector>
 3
 4 int main() {
 5     std::vector<int> numbers = {10, 20, 30, 40, 50};
 6     std::vector<int>::iterator it = numbers.begin();
 7     std::cout << *it << std::endl; // 10
 8     std::cout << *(it++) << std::endl;  // 10
 9     std::cout << *it << std::endl; // 20
10     std::cout << *(++it) << std::endl; // 30
11     std::cout << *it << std::endl; // 30
12 }
```

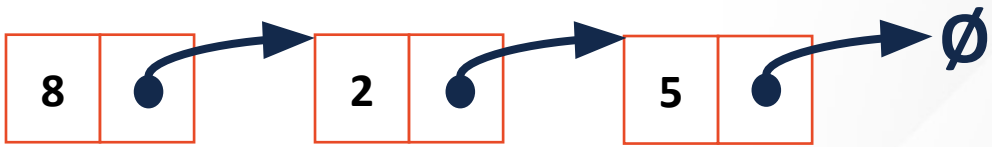
Defining Pre and Post Increment

```
1  // Pre-increment (++it)
2      Iterator& operator++() {
3          ++ptr;
4          return *this;
5      }
6
7  // Post-increment (it++)
8      Iterator operator++(int) {
9          Iterator temp = *this; // save current state
10         ++ptr;                 // advance iterator
11         return temp;           // return original
12     }
```

::begin	::end



::begin	::end
ArrayIterator(0)	ArrayIterator(size_)
ListIterator(head_)	ListIterator(nullptr)



Summary

1. An iterator behaves like a pointer through a container
2. It lets you traverse containers without knowing how they’re built
3. You need to implement begin(), end(), *(), ++(), and !=