

Vectors and Lists

- **Both are linear data structures**
  - Data elements arranged “in a line”
- **Primary difference**
  - Vector supports fast random access
  - List optimized for middle insert/delete
- **Let’s take a closer look**

---

---

---

---

---

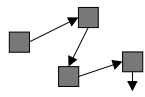
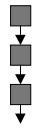
---

---

---

Sorting a Class (of Students)

- **Want in alphabetical order**
- **Sit in assigned seats?**
  - Make room for new student?
- **Sit anywhere, note predecessor in sequence?**
  - Identify nth student?




---

---

---

---

---

---

---

---

Vector/List Comparison

	Vector	List
Variable size	Yes	Yes
Sequential access	Yes	Yes
Insert at end	Yes	Yes
Delete at end	Yes	Yes
Insert in middle	Slow	Fast
Delete from middle	Slow	Fast
Random access	Yes	No

---

---

---

---

---

---

---

---

STL List Definition - Public

```

class list {
public:
    size_type size() const;
    void push_front(const T& x);
    void push_back(const T& x);
    void erase(iterator first,
               iterator last);
    void remove(const T& value);
};

```

"T" is the element type

---

---

---

---

---

---

---

---

---

---

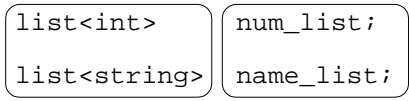
STL List - Initialization

Create an empty list of specified type

```

#include <list>
...

```



List type

List object name

---

---

---

---

---

---

---

---

---

---

STL List - Clear

Remove all elements from a list

```

list<int>    lis;

```

```

lis.clear();

```

clear member function

Could also use erase() with begin() and end().

---

---

---

---

---

---

---

---

---

---

STL List - Add Item

Add element at front or back of list  
list<int> nums;  
list<string> strs;  
string hi = "Hello";

```
nums.push_back (25);  
strs.push_front (hi);
```

Member functions

Values to add

---

---

---

---

---

---

---

---

STL List - Capacity, Size

Return current # of elements

```
list<int> nums;
```

```
(unsigned int) ss = nums.size();
```

Actual return type is  
list<int>::size\_type,  
but good enough?

Member function

---

---

---

---

---

---

---

---

STL List - Deleting by Value

Delete all elements with given value  
list<int> nums;

```
nums.push_back (25);  
nums.push_back (12);  
nums.push_back (-2);  
// 25 12 -2
```

Add to list

```
nums.remove (12);  
// 25 -2
```

Remove from list

---

---

---

---

---

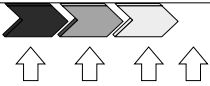
---

---

---

STL List Iterators

- **Separate objects**
  - Not part of list object
- **Specify element position**
  - Within one list (at a time)
- **Valid positions**
  - First element to "just beyond last"




---

---

---

---

---

---

---

---

Declaring STL Iterators

```

List types
list<int>      nums;
list<string>  strs;

list<int>::iterator  n_iter;
list<string>::iterator s_iter;
  
```

List iterator types

---

---

---

---

---

---

---

---

Simple Iterator Operators

- \* Element access
- ++ Next position
- == Comparison
- != Comparison

---

---

---

---

---

---

---

---

### STL Iterator Traversal



```

...
list<string> words;
list<string>::const_iterator iter;
...
unsigned int total = 0;
for (iter = words.begin();
     iter != words.end();
     iter++)
{
    total += (*iter).length();
}

```

Access element (note parentheses)      Initialize, test, move

---

---

---

---

---

---

---

---

### STL List Element Search

```

...
list<int> nums;
list<int>::iterator n_iter;
... // 7 10 3 10 15
n_iter = find (nums.begin(),
              nums.end(),
              10);
// n_iter at second element
if (n_iter != nums.end())
{
    nums.erase (n_iter);
} // 7 3 10 15

```

Search range for value

Search value must be same type as list elements!

---

---

---

---

---

---

---

---

### Other STL List Operations



- Access first, last element
- Insert element (middle)
- Delete first, last element
- Reverse list elements
- Sort list (member function!)
  - Using "<" operator on element type

---

---

---

---

---

---

---

---

STL List Benefits

- **Flexible container class**
  - Efficient insert, delete, grow
    - Lacks random access of vector
- **Separate iterator objects**
  - Multiple iterators on same list
  - Iterators may be const
    - For const list or reference argument
    - E.g., `list<int>::const_iterator iter;`

---

---

---

---

---

---

---

---

Design Exercise



- **Read integers from a file**
- **Store in an STL list**
- **Find minimum and maximum**
- **Delete min and max elements**
  - Delete all instances of min and max
- **Print remaining elements**

---

---

---

---

---

---

---

---