Object-Oriented Programming (OOP)
Lecture No. 41

Standard Template Library

- C++ programmers commonly use many data structures and algorithms
- So C++ standard committee added the STL to C++ standard library
- STL is designed to operate efficiently across many different applications

…Standard Template Library

- STL consists of three key components
  - Containers
  - Iterators
  - Algorithms

STL Promotes Reuse

- Software reuse saves development time and cost
- Once tested component can be reused several times

STL Containers

- Container is an object that contains a collection of data elements
- STL provides three kinds of containers
  - Sequence Containers
  - Associative Containers
  - Container Adapters

Sequence Containers

- A sequence organizes a finite set of objects, all of the same type, into a strictly linear arrangement

...Sequence Containers

- **vector**
  - Rapid insertions and deletions at back end
  - Random access to elements
- **deque**
  - Rapid insertions and deletions at front or back
  - Random access to elements
- **list**
  - Doubly linked list
  - Rapid insertions and deletions anywhere

Example – STL Vector

```
#include <vector>
int main() {
  std::vector< int > iv;
  int x, y;
  char ch;
  do {
     cout<<"Enter the first integer:";
     cin >> x;
     cout<<"Enter the second
                              integer:";
     cin >> y;
```

…Example – STL Vector

```
        iv.push_back( x );
        iv.push_back( y );
        cout << "Current capacity of iv =
        " << iv.capacity() << endl;
        cout << "Current size of iv ="
                 << iv.size() << endl;
        cout<<"Do you want to continue?";
        cin >> ch;
    } while ( ch == 'y' );
}
```

Sample Output

```
Enter the first integer: 1

Enter the second integer: 2

Current capacity of iv = 2

Current size of iv = 2

Do you want to continue? y
```

...Sample Output

```
Enter the first integer: 3

Enter the second integer: 4

Current capacity of iv = 4

Current size of iv = 4

Do you want to continue? y
```

...Sample Output

```
Enter the first integer: 5

Enter the second integer: 6

Current capacity of iv = 8

Current size of iv = 6

Do you want to continue? n
```

Example – STL Deque

```
#include <deque>

int main() {
  std::deque< int > dq;
  dq.push_front( 3 );
  dq.push_back( 5 );

  dq.pop_front();
  dq.pop_back()
  return 0;
}
```

Example – STL List

```
#include <list>
int main() {
  std::list< float > _list;
  _list.push_back( 7.8 );
  _list.push_back( 8.9 );
  std::list< float >::iterator it
                        = _list.begin();
  _list.insert( ++it, 5.3 );
  return 0;
}
```

Associative Containers

- An associative container provide fast retrieval of data based on keys

...Associative Containers

- **`set`**
  - No duplicates
- **`multiset`**
  - Duplicates allowed
- **`map`**
  - No duplicate keys
- **`multimap`**
  - Duplicate keys allowed

Example – STL Set

```
#include <set>
int main() {
  std::set< char > cs;
  cout << "Size before insertions: "
       << cs.size() << endl;
  cs.insert( 'a' );
  cs.insert( 'b' );
  cs.insert( 'b' );
  cout << "Size after insertions: "
       << cs.size();
  return 0;
}
```

Output

```
Size before insertions: 0

Size after insertions: 2
```

Example – STL Multi-Set

```
#include <set>
int main() {
  std::multiset< char > cms;
  cout << "Size before insertions: "
       << cms.size() << endl;
  cms.insert( 'a' );
  cms.insert( 'b' );
  cms.insert( 'b' );
  cout << "Size after insertions: "
       << cms.size();
  return 0;
}
```

Output

```
Size before insertions: 0

Size after insertions: 3
```

Example – STL Map

```
#include <map>
int main() {
  typedef std::map< int, char > MyMap;
  MyMap m;
  m.insert(MyMap::value_type(1, 'a'));
  m.insert(MyMap::value_type(2, 'b'));
  m.insert(MyMap::value_type(3, 'c'));
  MyMap::iterator it = m.find( 2 );
  cout << "Value @ key " << it->first
       << " is " << it->second;
  return 0;
}
```

Output

```
Value @ key 2 is b
```

Example – STL Multi-Map

```
#include <map>

int main() {
  typedef std::multimap< int, char >
                                    MyMap;
  MyMap m;
  m.insert(MyMap::value_type(1, 'a'));
  m.insert(MyMap::value_type(2, 'b'));
  m.insert(MyMap::value_type(3, 'b'));
```

...Example – STL Multi-Map

```
  MyMap::iterator it1 = m.find( 2 );
  MyMap::iterator it2 = m.find( 3 );
  cout << "Value @ key " << it1->first
     << " is " << it1->second << endl;
  cout << "Value @ key " << it2->first
     << " is " << it2->second << endl;
  return 0;
}
```

Output

```
Value @ key 2 is b

Value @ key 3 is b
```

First-class Containers

- Sequence and associative containers are collectively referred to as the first-class containers

Container Adapters

- A container adapter is a constrained version of some first-class container

...Container Adapters

- **`stack`**
  - Last in first out (LIFO)
  - Can adapt **`vector`**, **`deque`** or **`list`**
- **`queue`**
  - First in first out (FIFO)
  - Can adapt **`deque`** or **`list`**
- **`priority_queue`**
  - Always returns element with highest priority
  - Can adapt **`vector`** or **`deque`**

Common Functions for All Containers

- Default constructor
- Copy Constructor
- Destructor
- `empty()`
  - Returns true if container contains no elements
- `max_size()`
  - Returns the maximum number of elements

...Common Functions for All Containers

- `size()`
  - Return current number of elements
- `operator = ()`
  - Assigns one container instance to another
- `operator < ()`
  - Returns true if the first container is less than the second container

...Common Functions for All Containers

- `operator <= ()`
  - Returns true if the first container is less than or equal to the second container
- `operator > ()`
  - Returns true if the first container is greater than the second container
- `operator >= ()`
  - Returns true if the first container is greater than or equal to the second container

...Common Functions for All Containers

- `operator == ()`
  - Returns true if the first container is equal to the second container
- `operator != ()`
  - Returns true if the first container is not equal to the second container
- `swap ()`
  - swaps the elements of the two containers

Functions for First-class Containers

- `begin()`
  - Returns an `iterator` object that refers to the first element of the container
- `end()`
  - Returns an `iterator` object that refers to the next position beyond the last element of the container

...Functions for First-class Containers

- `rbegin()`
  - Returns an `iterator` object that refers to the last element of the container
- `rend()`
  - Returns an `iterator` object that refers to the position before the first element

...Functions for First-class Containers

- `erase( iterator )`
  - Removes an element pointed to by the `iterator`
- `erase( iterator, iterator )`
  - Removes the range of elements specified by the first and the second `iterator` parameters

...Functions for First-class Containers

- `clear()`
  - erases all elements from the container

Container Requirements

- Each container requires element type to provide a minimum set of functionality e.g.
- When an element is inserted into a container, a copy of that element is made
  - Copy Constructor
  - Assignment Operator

…Container Requirements

- Associative containers and many algorithms compare elements
  - Operator ==
  - Operator <