159.234 Lecture 23

- More on the Standard Template Library
- A guided tour with common examples of the containers

More on the Standard Template Library (STL)

STL has:

- bitset — a set of bits
- deque — a double ended-queue
- list — a linear list
- map — key value pairs (unique association)
- multimap — key-value pairs (multiple associations)
- multiset — set (elements need not be unique)
- priority_queue — a priority based queue
- queue — a queue
- set — a set in which elements are unique
- stack — a stack
- vector — a dynamic array

bitset<24> bs;

```
#include <iostream>
#include <bitset>
using namespace std;
int main(){
    const int N = 24;
    bitset<N> bs;
    bs.reset();
    bs.set(1); bs.set(2);  bs.set(4); bs.set(8);
    cout << bs[8] << endl;
    for(int i=N-1;i>=0;i--){
      cout << bs.test(i);
      if (! ( i%8 ) ) cout << " ";
    } cout << endl;

    bs <<= 2 ;
    for(int i=N-1;i>=0;i--){
      cout << bs.test(i);
      if (! ( i%8 ) ) cout << " ";
    }  cout << endl;}
}
```

Output is:

```
% a.out
1
00000000 00000001 00010110
00000000 00000100 01011000
```

deque<int> dq;

```
deque<int> dq;
dq.push_back( 4 );
dq.push_back( 5 );
dq.push_front( 3 );
dq.push_front( 2 );
dq.push_front( 1 );
cout << "Max  size of  deque is " << dq.max_size() << endl;
cout << "Element 2 is " << dq[2] << endl;
for(deque<int>::iterator it=dq.begin();it!=dq.end();it++){
  cout << *it << " ";
} cout << endl;
deque<int>::iterator it=dq.begin();
it += 3;
dq.erase( it );
for(deque<int>::reverse_iterator
      rit=dq.rbegin();rit!=dq.rend();rit++){
  cout << *rit << " ";
} cout << endl;
deque<int> dq2;
dq2.push_front( 30 );
dq2.push_front( 20 );
dq2.push_front( 10 );
dq.swap( dq2 );
for(deque<int>::iterator it=dq.begin();it!=dq.end();it++){
  cout << *it << " ";
} cout << endl;
```

Output:

% a.out

Max size of deque is 4294967295

Element 2 is 3

1 2 3 4 5

5 3 2 1

10 20 30

list <int> l1, l2;

```
list<int> l1, l2;
l1.push_front(2);l1.push_front(4);l1.push_front(6);l1.push_front(8);
l2.push_back(1);l2.push_back(3);l2.push_back(5); l2.push_back(7);

for(list<int>::iterator it = l1.begin(); it != l1.end(); it++){
  cout << *it << " ";
} cout << endl;

l1.sort();

for(list<int>::iterator it = l1.begin(); it != l1.end(); it++){
  cout << *it << " ";
} cout << endl;

for(list<int>::iterator it = l2.begin(); it != l2.end(); it++){
  cout << *it << " ";
} cout << endl;

l1.merge( l2 );

for(list<int>::iterator it = l1.begin(); it != l1.end(); it++){
  cout << *it << " ";
} cout << endl;
```

Output:

```
% a.out
8 6 4 2
2 4 6 8
1 3 5 7
1 2 3 4 5 6 7 8
```

map<string,int> m1;

```
map<string,int> m1, m2;
map<string,int>::iterator  iter;  // points to a pair<string,int>

string one("One");
string two("Two");
string three("Three");
string four("Four");
string five("Five");

m1[one]   = 1;
m2[two]   = 2;
m1[three] = 3;
m2[four]  = 4;
m1[five]  = 5;

for(iter=m1.begin();iter!=m1.end();iter++){
  cout << "Found " << iter->second << " keyed by " << iter->first <<
      endl;
} cout << endl;

m1.swap(m2);

for(iter=m1.begin();iter!=m1.end();iter++){
  cout << "Found " << iter->second << " keyed by " << iter->first <<
      endl;
}
```

NB map's iterator is pointer to a pair - deref using members **first** and **second**

Output:

```
% a.out
Found 5 keyed by Five
Found 1 keyed by One
Found 3 keyed by Three

Found 4 keyed by Four
Found 2 keyed by Two
```

multimap<string,int> m1, m2;

```
multimap<string,int> m1, m2;
multimap<string,int>::iterator  iter;  // points to a pair<string,int>

string one("One");  string two("Two");  string three("Three");
string four("Four");string five("Five");string six("Six");

m1.insert( m1.end(), make_pair(one, 1) );  // cannot use [] notation
m1.insert( m1.end(), make_pair(two, 2) );
m1.insert( m1.end(), make_pair(three, 3) );
m1.insert( m1.end(), make_pair(four, 4) );
m1.insert( m1.end(), make_pair(five, 5) );
m1.insert( m1.begin(), make_pair(six, 6) );
m1.insert( m1.begin(), make_pair(four, 40) );

for(iter=m1.begin();iter!=m1.end();iter++){
  cout << "Found " << iter->second << " keyed by " << iter->first <<
      endl;
} cout << endl;

m1.swap(m2);
for(iter=m1.begin();iter!=m1.end();iter++){
  cout << "Found " << iter->second << " keyed by " << iter->first <<
      endl;
}
```

NB #include<map> not <multimap>

Output:

% a.out

Found 5 keyed by Five
Found 4 keyed by Four
Found 40 keyed by Four
Found 1 keyed by One
Found 6 keyed by Six
Found 3 keyed by Three
Found 2 keyed by Two

multiset<int> ms;

```
multiset<int> ms;
multiset<int>::iterator  iter;  // points to a pair<string,int>

ms.insert( ms.end(), 1 );  // cannot use [] notation
ms.insert( ms.end(), 3);
ms.insert( ms.end(), 2);
ms.insert( ms.end(), 4);
ms.insert( ms.end(), 4);
ms.insert( ms.end(), 5);
cout << "Multiset contains: ";
for(iter=ms.begin();iter!=ms.end();iter++){
  cout << " " << *iter;
} cout << endl;

iter = ms.find(3);
cout << "Found " << *iter++ << " followed by " << *iter << endl;
iter = ms.find(4);
cout << "Found " << *iter++ << " followed by " << *iter << endl;

ms.erase(iter);

cout << "Multiset now contains: ";
for(iter=ms.begin();iter!=ms.end();iter++){
  cout << " " << *iter;
} cout << endl;
```

NB #include <set> not multiset

Output:

```
% a.out
Multiset contains:  1 2 3 4 4 5
Found 3 followed by 4
Found 4 followed by 4
Multiset now contains:  1 2 3 4 5
```

queue<int> q;

```
#include <iostream>
#include <queue>

using namespace std;

int main(){

  queue<int> q;   // has no iterator

  for(int i=0; i < 6 ;i++){
    q.push(i);
  }

  cout << "Queue size " << q.size() << endl;
  cout << "Queue empty ? " << boolalpha << q.empty() << endl;

  cout << "Queue contains:";
  while( !q.empty() ){
    cout << " " << q.front();

    q.pop();  // discards head of queue

  } cout << endl;

  cout << "Queue size " << q.size() << endl;
}
```

Output:

% a.out
Queue size 6
Queue empty ? false
Queue contains: 0 1 2 3 4 5
Queue size 0

set <int> s1, s2, s3;

```
set<int> s1, s2, s3;
set<int>::iterator it;
for(int i=9;i>=0;i--){
  if( i%2 )
    s1.insert(s1.begin(), i);
  else
    s2.insert(s2.end(), i);
}
cout << "size of s1 is " << s1.size() << endl;
cout << "size of s2 is " << s2.size() << endl;
cout << "s1 contains";
for(it=s1.begin();it != s1.end(); it++){
  cout << " " << *it;
} cout << endl;
cout << "s2 contains";
for(it=s2.begin();it != s2.end(); it++){
  cout << " " << *it;
} cout << endl;
s3 = s1;
s3.insert( s2.begin(), s2.end() );
cout << "s3 contains";
for(it=s3.begin();it != s3.end(); it++){
  cout << " " << *it;
} cout << endl;
if( s3.find(4) != s3.end() )
  cout << "s3 contains  4" << endl;
else
  cout << "s3 does not contain 4" << endl;
if( s3.find(11) != s3.end() )
  cout << "s3 contains  11" << endl;
else
  cout << "s3 does not contain 11" << endl;
```

Output:

% a.out

size of s1 is 5

size of s2 is 5

s1 contains 1 3 5 7 9

s2 contains 0 2 4 6 8

s3 contains 0 1 2 3 4 5 6 7 8 9

s3 contains 4

s3 does not contain 11

priority_queue<int> q;

```
#include <iostream>
#include <queue>
#include <functional>
#include <vector>
using namespace std;

int main(){
// priority_queue<int> q;   // has no iterator

  priority_queue<int, vector<int>, less<int> > q;
    // by default  vector is  used as container
    // by default  less is  used as comparator

  for(int i=0; i < 6 ;i++){
   q.push(i);
  }
  cout << "Queue size " << q.size() << endl;
  cout << "Queue empty ? " << boolalpha << q.empty() << endl;
  cout << "Priority element is " << q.top() << endl;
  cout << "Queue priority elements, in order, are:";
  while( !q.empty() ){
   cout << " " << q.top();
   q.pop();  // discards head of queue
  } cout << endl;
  cout << "Queue size " << q.size() << endl;
}
```

NB #include <queue> not priority_queue

Output:

% a.out
Queue size 6
Queue empty ? false
Priority element is 5
Queue priority elements, in order, are: 5 4 3 2 1 0
Queue size 0

Output (with **greater**):

% a.out
Queue size 6
Queue empty ? false
Priority element is 0
Queue priority elements, in order, are: 0 1 2 3 4 5
Queue size 0

stack<int> s;

```
#include <iostream>
#include <stack>
using namespace std;

int main(){
  stack<int> s;   // has no iterator

  cout << "Pushing onto stack: ";
  for(int i=0; i < 6 ;i++){
    cout << " " << i;
    s.push(i);
  }
  for(int i=12; i >= 6 ;i--){
    cout << " " << i;
    s.push(i);
  }
  cout << endl;
  cout << "Stack size " << s.size() << endl;
  cout << "    Top of stack is:";
  while( !s.empty() ){
    cout << " " << s.top();
    s.pop();  // discards top of stack
  } cout << endl;
  cout << "Stack size " << s.size() << endl;
}
```

Output:

% a.out

Pushing onto stack: 0 1 2 3 4 5 12 11 10 9 8 7 6

Stack size 13

Top of stack is: 6 7 8 9 10 11 12 5 4 3 2 1 0

Stack size 0

vector<int> v;

```
#include <iostream>
#include <vector>
using namespace std;

int main(){
  vector<int> v;
  vector<int>::iterator it, start, end;
  for(int i=0;i<10;i++){
    v.push_back(i);
  }
  cout << "v size = " << v.size() << endl;
  start = v.begin();
  start += 3;   // now points  at element [3]
  end = start;
  end += 4;     // now points  at  element [7]
  v.erase(start, end );  // erases 3,4,5,6
  cout << "v size = " << v.size() << endl;
  for(it=v.begin();it<v.end();it++){
    *it *= 10;
  }
  cout  << "v contains: ";
  for(int i=0;i<v.size(); i++){
    cout << " " << v[i];
  } cout << endl;

}
```

Output:

% a.out

v size = 10

v size = 6

v contains: 0 10 20 70 80 90

STL Container Summary

- STL provides useful containers
- Can be parameterised by a contained class
- iterators useful
- Not all have iterators

- See test-stl.zip

- See “The Complete Reference C++” Fourth Edition, Herbert Schildt, McGraw Hill, ISBN0-07-222680-3, Chaper 24 and Chapter 33.