Lecture fourteen

Introducing the Standard Template Library

Ref: Herbert Schildt, Teach Yourself C++, Third Edn (Chapter 14)

© Dr. M. Mahfuzul Islam
Professor, Dept. of CSE, BUET

Standard Template Library (STL)

- **Three foundation Items: containers, algorithms and iterators.**
- **Containers: Containers are objects that hold other objects. Different types of Containers:**
  - **vector: defines a dynamic array.**
  - **queue: creates a queue.**
  - **list: provides a linear list.**
  - **map: defines a map that provides access to values with unique keys.**
- **Algorithms: Algorithms act on containers like initialization, sorting, searching, transforming the contents, etc.**
- **Iterators: Iterators are objects that work like pointers to the contents of a containers with the ability to cycle.**

Standard Template Library (STL)

Five Iterators:

Random access, Bidirectional, Forward, input and output.

Five Reverse Iterators:

Bilter, Forlter, Inlter, Outlter, Randlter.

- **allocator: Each container has an allocator that manage memory allocation for containers.**
- **predicate: Some containers and algorithms use special type of function called predicate. Two variations of predicate: UnPred- accept one argument and BinPred- accepts two arguments: first, second.**

The standard C++ library includes two headers <utility> and <functional>

The Container Classes

<bitset>, <deque>, <list>, <map>, <set>, <queue>, <stack>, <vector>

Vector:

- The most general purpose container. Vector class supports dynamic array.
- Some most important member functions are: size(), begin(), end(), push_back(), insert(), erase().

The Container Classes

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

int main(){
    vector<int> v;
    int i;

    cout <<"Size: "<< v.size() << endl;
     for(i = 0; i<10; ++i) v.push_back(i);

    cout <<"Size: "<< v.size() << endl;
    for(i=0; i<v.size(); ++i)
          cout << v[i] << " ";

    for(i= 0; i<5; ++i) v.push_back(i+10);

    cout <<"Size: "<< v.size() << endl;
    vector<int>::iterator p = v.begin();

    while(p != v.end()){
         cout << *p <<"  ";
         ++p;
     }

         p = v.begin();
         p += 2;

         v.erase(p, p+10);
         for(i=0; i<v.size(); ++i)
               cout << v[i] << " ";

          p = v.begin();
          p += 2;

          v. insert(p, 10, 9);
          for(i=0; i<v.size(); ++i)
                cout << v(i) << " ";

          return 0;
}
```

OUTPUT:

Size: 0
Size: 10
0 1 2 3 4 5 6 7 8 9
Size: 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
0 1 12 13 14
0 1 9 9 9 9 9 9 9 9 12 13 14

The Container Classes

Lists:

- The list class supports a bidirectional, linear list. Unlike a vector, list does not support random access. Access can be either front to back or back to front.
- Some most important member functions are: size(), begin(), end(), push_back(), push_front(), pop_back(), pop_front(), empty(), sort(), merge().
- merge() member function merge one ordered list with another ordered list which remains with invoking list. The second list becomes empty.

lst1: ACEGI
lst2: BDFHJ
After execution of lst1.merge(lst2);
lst1: ABCDEFGHIJ
lst2:

The Container Classes

Maps:

- The map class supports an associative container in which unique keys are mapped with values.
- The value in first contains the key and the value in second contains the value.
- The key/value pair stored are:

A	0
B	1
C	2

The program for the above pair:

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

int main(){
    map<char, int> m;
    int i;

    for(i=0; i<10; ++i)
        m.insert(pair<char,int>('A'+i, i));

    char ch;
    cout << "Enter key:";
    cin >> ch;
    map<char, int>:: iterator p;

    p = m.find(ch);
    if (p !=m.end())
        cout <<  p->second;
    else
        cout << "key is not in map\n";

    return 0;
}
```

- The function make_pair() can be used instead of pair.

```
m.insert(make_pair((char) ('A'+i), i));
```

Algorithms

- Algorithms act on containers. Header <algorithm> should be included.
- Some algorithms:
  - int n = count(v.begin(), v.end(), 1): returns the number of elements in the sequence beginning at start (v.begin()) and ending at end (v.end()) that match val.
  - int n = count_if(v.begin(), v.end(), even): returns the number of elements in the sequence beginning at start (v.begin()) and ending at end (v.end()) for which the unary predicate pfn (even) returns true.
  - Remove_copy(v.begin(), v.end(), v2.begin(), 1): copies elements from the specified range that are not equal to val (1) and puts the result into the sequence pointed by result (v2.begin()).
  - reverse(v.begin(), v.end()): reverses the order of the range specified.
  - transform(x.begin(), x.end(), x.begin(), xform): modifies each element in a range according to a function.

Algorithms

```
#include <iostream>
#include <list>
#include <algorithm>
using namespace std;

int xform(int i){
      return i*i;
}

int main(){
      list< int> x;
      int i;

      for(i=0; i<10; ++i) x.push_back(i);
      list<int>::iterator p = x.begin();

      while( p != x.end()){
            cout << p << " ";  ++p
      }

      p = transform(x.begin(), x.end(), x.begin(), xform);

      p = x.begin();
      while( p != x.end()){
            cout << p << " ";  ++p
      }

      return 0;
}
```