STANDARD TEMPLATE LIBRARY

STANDARD TEMPLATE LIBRARY

- *Containers*
  - *Vector*
  - *List*
  - ***Statck***
  - ***Queue***
  - ***…***
- *Iterators:* To cycle through the contents of a container.
- *Algorithms: They* act on containers.
  - To manipulate the contents of containers.
  - Their capabilities include initialization, sorting, searching, and transforming the contents of containers.

Example-1 : vector – A dynamic array

```
4  int main()
5  {
6      vector<int> v;
7      cout<<v.capacity();
8      v.push_back(11);
9      cout<<"\n capacity="<<v.capacity();
10     v.push_back(22);
11     cout<<"\n capacity="<<v.capacity();
12     v.push_back(33);
13     cout<<"\n capacity="<<v.capacity();
14     v.push_back(34);  v.push_back(35);  v.push_back(36);  v.push_back(37);
15     cout<<"\n capacity="<<v.capacity();
16     v.push_back(38); v.push_back(39);
17     cout<<"\n capacity="<<v.capacity();
18 }
```

```
0
 capacity=1
 capacity=2
 capacity=4
 capacity=8
 capacity=16
```

Example-2: vector operations

```
2  #include<vector>
3  using namespace std;
4  int main()
5  {
6      vector<int> v;
7      cout<<v.size();
8      for( int i = 1; i <= 5; i++ )
9          v.push_back( i * 10 );
10     cout<<"\n vector size="<<v.size()<<endl;
11     v.insert(v.begin()+2,25);
12     v.erase(v.begin()+4);
13     v.push_back(60);
14     v.push_back(70);
15     v[3] = 100;  //v.at(3) = 100;
16     v.pop_back();
17     for( int i = 0; i < v.size(); i++ )
18         cout<<v[i]<<"  ";
19 }
```

OUTPUT

```
0
 vector size=5
10   20   25   100   50   60
```

Example-3: iterators:

```
The vector contents:
a b c
```

```
 4  int main()
 5  {
 6      vector<char> v;
 7      vector<char>::iterator it; // create an iterator
 8      int i;
 9      for( int i = 0 ; i < 3 ; i ++ )
10          v.push_back( 'a' + i );
11          // display contents of vector
12      cout << "The vector contents:\n";
13      for( it = v.begin(); it != v.end() ; it++ )
14          cout << *it << " ";
15  }
```

Lists

- *The **list** class supports a bidirectional, linear list.*
- *Unlike a vector, which supports random access, a list can be accessed sequentially only.*
- *Since lists are bidirectional, they may be accessed front to back or back to front.*

Example-4: List operations:

```
10 20 30 40
```

```
6  int main()
7  {
8      list<int> ilist;
9
10     ilist.push_back(30);              // push items on back
11     ilist.push_back(40);
12     ilist.push_front(20);             // push items on front
13     ilist.push_front(10);
14
15     int size = ilist.size();          // number of items
16
17     for(int j=0; j<size; j++)
18     {
19         cout << ilist.front() << ' ';  // read item from front
20         ilist.pop_front();             // pop item off front
21     }
22     cout << endl;
23 }
```

Example-5: List with iterator:

```
4  int main()
5  {
6      list<int> lst; // create an empty list
7      int i;
8      for(i=0; i<10; i++)
9          lst.push_back(i);
10     cout << "Size = " << lst.size() << endl;
11
12     list<int>::iterator p;
13     for( p = lst.begin(); p != lst.end() ; p++ )
14         *p = *p + 100;
15     cout << "Contents: ";
16     for( p = lst.begin(); p != lst.end() ; p++ )
17         cout<< *p <<"  ";
18 }
```

```
Size = 10
Contents: 100  101  102  103  104  105  106  107  108  109
```

Example-6: List with reverse_iterator:

```
3  #include <list>
4  using namespace std;
5  int main()
6  {
7     int arr[] = { 2, 4, 6, 8, 10 };          // array of ints
8     list<int> theList;
9
10    for(int j=0; j<5; j++)                   // transfer array
11       theList.push_back( arr[j] );          // to list
12
13    list<int>::reverse_iterator revit;       // reverse iterator
14
15    revit = theList.rbegin();                // iterate backwards
16    while( revit != theList.rend() )         // through list,
17       cout << *revit++ << ' ';              // displaying output
18    cout << endl;
19 }
```

Example-7: List insertion at both ends:

```
List1 : 0 1 2 3 4 5 6 7 8 9
List2 : 9 8 7 6 5 4 3 2 1 0
```

```
4  int main()
5  {
6      list<int> lst1, lst2;
7      int i;
8      for(i=0; i<10; i++)
9          lst1.push_back( i );
10     for(i=0; i<10; i++)
11         lst2.push_front( i );
12     list<int>::iterator it;
13
14     // Display the content of lst1, lst2
15     cout<<"\n List1 : ";
16     for(it=lst1.begin();it !=lst1.end();it++)
17         cout<<*it<<" ";
18
19     cout<<"\n List2 : ";
20     for(it=lst2.begin();it !=lst2.end();it++)
21         cout<<*it<<" ";
22 }
```

Example-8: algorithm

```cpp
// sorts an array of integers
#include <iostream>
#include <algorithm>
using namespace std;

                                    // array of numbers
int arr[] = {45, 2, 22, -17, 0, -30, 25, 55};

int main()
{
    sort(arr, arr+8);               // sort the numbers

    for(int j=0; j<8; j++)          // display sorted array
       cout << arr[j] << ' ';
    cout << endl;
    return 0;
}
```

```
-30 -17 0 2 22 25 45 55
```

Example-9: algorithm with vector

```
10 20 30 40
-600 100 200
```

```
5  int main()
6  {
7      vector<int> v1;
8
9      for( int i = 4 ; i > 0 ; i-- )
10         v1.push_back( i* 10 );
11
12     reverse( v1.begin(), v1.end() );
13
14     vector<int> v2;
15     v2.push_back( 100 ); v2.push_back( -600 ); v2.push_back( 200 );
16
17     sort( v2.begin(), v2.end() );
18     // Display v1 , v2
19     for( int i = 0 ; i < 4 ; i++ ) cout<<v1[i]<<" "; cout<<endl;
20     for( int i = 0 ; i < 3 ; i++ ) cout<<v2[i]<<" ";
21 }
```

Example-10 : Algorithm: sort() with array

```cpp
// sorts an array of integers
#include <iostream>
#include <algorithm>
using namespace std;

int arr[] = {45, 2, 22, -17, 0, -30, 25, 55};   // array of numbers

int main()
{
   sort(arr, arr+8);                // sort the numbers

   for(int j=0; j<8; j++)           // display sorted array
      cout << arr[j] << ' ';
   cout << endl;
   return 0;
}
```

```
-30 -17 0 2 22 25 45 55
```

Example-11 : Algorithm: merge()

```
5  int main()
6  {
7      list<int> lst1;
8      for(int i = 1 ; i <= 3 ; i++ )
9          lst1.push_back( i*10 );
10     lst1.push_front( 30 );
11     list<int>::iterator it;
12     int freq = count( lst1.begin(), lst1.end(), 30 );
13     cout<<"\nThe element 30 appears "<<freq<<" times.\n";
14     lst1.pop_front();
15
16     for( it = lst1.begin() ; it != lst1.end() ; it++ )
17         cout<< *it <<" ";
18
19     list<int> lst2;
20     lst2.push_back( 11 ); lst2.push_back( 22 ); lst2.push_back( 33 );
21     lst1.merge( lst2 );       cout<<endl;
22     for( it = lst1.begin() ; it != lst1.end() ; it++ )
23         cout<< *it <<" ";
24 }
```

```
The element 30 appears 2 times.
10 20 30
10 11 20 22 30 33
```