standard template library (STL) list container implement following member functions of list class: empty, insert, reverse, sort, Unique, using iterator.

**Aim:**

To Using understand concept of standard template library (STL).

**Description:**

We can implement program Using standard template library (STL) list container implement following member functions of list class: empty, insert, reverse, sort, Unique, using iterator.

**Oop Concepts Used:**

1. **Standard Template Library (STL):-**

Lists are sequence containers that allow non-contiguous memory allocation. As compared to vector, list has slow traversal, but once a position has been found, insertion and deletion are quick. Normally, when we say a List, we talk about doubly linked list. For implementing a singly linked list, we use forward list.

**2.List:-**

* List is a popularly used sequence container. Container is an object that holds data of same type. List container is implemented as doubly linked-list, hence it provides bidirectional sequential access to it's data.
* List doesn't provide fast random access, it only supports sequential access in both directions. List allows insertion and deletion operation anywhere within a sequence in constant time.
* Elements of list can be scattered in different chunks of memory. Container stores necessary information to allow sequential access to it's data. Lists can shrink or expand as needed from both ends at run time. The storage requirement is fulfilled automatically by internal allocator.
* Zero sized lists are also valid. In that case list.begin() and list.end() points to same location. But behavior of calling front() or back() is undefined.

**Syntax:-**

***List<type>object\_name;***

**Program:**

**#include<iostream>**

**#include<list>**

**#include<stdlib.h>**

**using namespace std;**

**void display(list<int> &l)**

**{**

**list<int>::iterator itr=l.begin();**

**cout<<endl;**

**for(itr=l.begin(); itr!=l.end(); itr++)**

**{**

**cout<<\*itr<<"\t";**

**}**

**}**

**void ins(list<int> &l)**

**{**

**int val;**

**cout<<"Enter value to insert : ";**

**cin>>val;**

**l.push\_back(val);**

**//l.insert(l.end(),val);**

**}**

**int main()**

**{**

**list<int> l1;**

**list<int> l2;**

**int value;**

**cout<<"Enter size of both lists : ";**

**cin>>value;**

**cout<<"List 1 Element : ";**

**for(int i=0; i<value; i++)**

**{**

**int ele1;**

**cin>>ele1;**

**l1.push\_back(ele1);**

**}**

**cout<<"List 2 Element : ";**

**for(int i=0; i<value; i++)**

**{**

**int ele2;**

**cin>>ele2;**

**l2.push\_back(ele2);**

**}**

**int choice;**

**do**

**{**

**cout<<endl;**

**cout<<"1.Display\n2.Sort\n3.Empty\n4.Insert\n5.Reverse\n6.Unique\n7.Exit\nChoice>> ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1: display(l1); display(l2);**

**break;**

**case 2: l1.sort(); l2.sort();**

**break;**

**case 3: if(l1.empty())**

**cout<<"List 1 is empty\n";**

**else**

**cout<<"List 1 is not empty\n";**

**if(l2.empty())**

**cout<<"List 2 is empty\n";**

**else**

**cout<<"List 2 is not empty\n";**

**break;**

**case 4:**

**int ans;**

**cout<<"In which list you want to enter:";**

**cin>>ans;**

**if(ans==1)**

**ins(l1);**

**else if(ans==2)**

**ins(l2);**

**break;**

**case 5: int ans2;**

**cout<<"In which list you want to enter:";**

**cin>>ans2;**

**if(ans2==1)**

**l1.reverse();**

**else if(ans2==2)**

**l2.reverse();**

**break;**

**case 6: l1.unique(); l2.unique();**

**break;**

**case 7: exit(0);**

**}**

**}while(choice!=7);**

**return 0;**

**}**

**Conclusion:**

Thus we have successfully done list by concept of list and we also used in build function to perform operation on list concept in oops.