C, C++, DSA in depth

list

Saurabh Shukla (MySirG)

Agenda

1. list
2. Creating list object
3. Accessing list elements
4. Implicit and explicit iterators
5. list methods

list

- The list class is a sequential container
- list is based on doubly linked list
- The header required is list
- list provides bidirectional iterator

How to create list Object ?

```
list <int> l1;
list <int> l2 = { 10, 30, 50 };
```

Accessing list elements

You can access list in the variety of ways.

1. implicit iterator
2. explicit iterator

Implicit Iterator

```
list <int> l1 = {10, 20, 30, 40};

for ( int x : l1 )
    cout<< x <<" ";
```

or

```
for (auto x : l1 )
    cout << x <<" ";
```

Explicit iterator

You can get an iterator object from the following members

①	begin()	end()	iterator
②	cbegin()	cend()	const_iterator
③	rbegin()	rend()	reverse_iterator
④	crbegin()	crend()	const_reverse_iterator

Explicit Iterator

```
list <int> l1 = { 50, 40, 10, 70, 60 };
list <int> :: iterator it;

for (it = l1.begin(); it != l1.end(); it++)
        cout << *it << " ";

list <int> :: const_iterator it;

for (it = l1.cbegin(); it != l1.cend(); it++)
        cout << *it << " ";
```

Explicit Iterator

```
list <int> l1 = { 50, 40, 10, 70, 60 };
list <int> :: reverse_iterator it;
for (it = l1.rbegin(); it != l1.rend(); it++)
        cout << *it << " ";

list <int> :: const_reverse_iterator it;
for (it = l1.crbegin(); it != l1.crend(); it++)
        cout << *it << " ";
```

Methods of list class

back()	returns last element
front()	returns first element
empty()	returns true or false
size()	returns the number of elements
swap()	swap elements of two lists
clear()	erases all the elements
merge()	removes the elements from the argument list, insert them into the target list and orders the new combined set of elements in ascending order or in some other specific order.

push_back()

pop_back()
push_front()
pop_front()
erase()
emplace()
emplace_back()
emplace_front()

insert() insert one or many elements

reverse()
remove()

insert single element

```
l1.insert(it, element);
```

insert same element multiple times

```
l1.insert(it, frequency, element);
```

insert few elements

```
l1.insert(it, {e1, e2, e3, ...});
```

add value at front

```
l1.push_front(element);
```

add value at last.

```
l1.push_back(element);
```

erase

Erase an element of vector

```
l1.erase(it);
```

Erase range of elements of vector

```
l1.erase(it1, it2)
```

it1: inclusive

it2: exclusive

Erase front and back element

```
l1.pop_front();
l1.pop_back();
```