C, C++, DSA in depth

forward_list

Saurabh Shukla (MySirG)

Agenda

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

forward_list

- The forward_list class is a sequential container
- forward_list is based on Singly linked list
- The header required is forward_list
- list provides forward iterator

How to create forward_list Object?

```
forward_list<int> l1;
forward_list<int> l2= {15, 30, 68, 42, 28};

l1.assign({6,12,18});
l1.assign( 3,20);   // 20, 20, 20
```

Accessing forward_list elements

You can access forward_list in the variety of ways.

1. implicit iterator
2. explicit iterator

Implicit Iterator

```
forward_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
⑤	before_begin()		
⑥	cbefore_begin()		

Explicit Iterator

```
forward_list <int> l1 = { 50, 40, 10, 70, 60 };
forward_list <int> :: iterator it ;
   for (it = l1.begin() ; it != l1.end() ; it++)
          cout << *it << " " ;
forward_list <int> :: const_iterator it ;
   for (it = l1.cbegin() ; it != l1.cend() ; it++)
          cout << *it << " " ;
```

Methods of forward_list

clear()
empty()
front()
insert_after()
emplace_after()
erase_after()
push_front()
emplace_front()
swap()
merge()
reverse()
sort()

push_front()
pop_front()
erase()
emplace()

emplace_front()
insert() insert one or many elements
reverse()