

STL

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 | <code>begin()</code> | <code>end()</code> | iterator |
| ② | <code>cbegin()</code> | <code>cend()</code> | const_iterator |
| ③ | <code>xbegin()</code> | <code>xend()</code> | reverse_iterator |
| ④ | <code>crbegin()</code> | <code>crend()</code> | const_reverse_iterator |
| ⑤ | <code>before_begin()</code> | | |
| ⑥ | <code>cbefore_begin()</code> | | |

forward-list

it ↓



Explicit Iterator

```
forward_list <int> ll = { 50, 40, 10, 70, 60 };
```

```
forward_list <int> :: iterator it;
```

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

```
forward_list <int> :: const_iterator it;
```

```
for (it = ll.cbegin(); it != ll.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()

`pop_front()`

`emplace_front()`