Forward List in C++ | Set 1 (Introduction and Important Functions)

Forward list in STL implements singly linked list. Introduced from C++11, forward list are useful than other containers in insertion, removal and moving operations (like sort) and allows time constant insertion and removal of elements.

It differs from list by the fact that forward list keeps track of location of only next element while list keeps track to both next and previous elements, thus increasing the storage space required to store each element. The drawback of forward list is that it cannot be iterated backwards and its individual elements cannot be accessed directly.

Forward List is preferred over list when only forward traversal is required (same as singly linked list is preferred over doubly linked list) as we can save space. Some example cases are, chaining in hashing, adjacency list representation of graph, etc.

Operations on Forward List :

1. assign() :- This function is used to assign values to forward list, its another variant is used to assign repeated elements.

```
// C++ code to demonstrate forward list
// and assign()
#include<iostream>
#include<forward_list>
using namespace std;

int main()
{
    // Declaring forward list
    forward_list<int> flist1;

    // Declaring another forward list
    forward_list<int> flist2;

    // Assigning values using assign()
    flist1.assign({1, 2, 3});

    // Assigning repeating values using assign()
    // 5 elements with value 10
    flist2.assign(5, 10);

    // Displaying forward lists
    cout << "The elements of first forward list are : ";
    for (int&a : flist1)
        cout << a << " ";
    cout << endl;

    cout << "The elements of second forward list are : ";
    for (int&b : flist2)
        cout << b << " ";
    cout << endl;

    return 0;
}
```

Output:

```
The elements of first forward list are : 1 2 3
The elements of second forward list are : 10 10 10 10 10
```

2. push_front() :- This function is used to insert the element at the first position on forward list. The value from this function is copied to the space before first element in the container. The size of forward list increases by 1.

3. emplace_front() :- This function is similar to the previous function but in this no copying operation occurs, the element is created directly at the memory before the first element of the forward list.

4. pop_front() :- This function is used to delete the first element of list.

```
// C++ code to demonstrate working of
// push_front(), emplace_front() and pop_front()
#include<iostream>
#include<forward_list>
using namespace std;

int main()
{
    // Initializing forward list
    forward_list<int> flist = {10, 20, 30, 40, 50};

    // Inserting value using push_front()
    // Inserts 60 at front
    flist.push_front(60);

    // Displaying the forward list
    cout << "The forward list after push_front operation : ";
    for (int&c : flist)
        cout << c << " ";
    cout << endl;

    // Inserting value using emplace_front()
    // Inserts 70 at front
    flist.emplace_front(70);

    // Displaying the forward list
    cout << "The forward list after emplace_front operation : ";
    for (int&c : flist)
       cout << c << " ";
    cout << endl;

    // Deleting first value using pop_front()
    // Pops 70
    flist.pop_front();

    // Displaying the forward list
    cout << "The forward list after pop_front operation : ";
    for (int&c : flist)
        cout << c << " ";
    cout << endl;

    return 0;
}
```

Output:

```
The forward list after push_front operation : 60 10 20 30 40 50
The forward list after emplace_front operation : 70 60 10 20 30 40 50
The forward list after pop_front operation : 60 10 20 30 40 50
```

4. insert_after() This function gives us a choice to insert elements at any position in forward list. The arguments in this function are copied at the desired position.

5. emplace_after() This function also does the same operation as above function but the elements are directly made without any copy operation.

6. erase_after() This function is used to erase elements from a particular position in the forward list.

```
// C++ code to demonstrate working of
// insert_after(), emplace_after() and erase_after()
#include<iostream>
#include<forward_list>
using namespace std;

int main()
{
    // Initializing forward list
    forward_list<int> flist = {10, 20, 30} ;

    // Declaring a forward list iterator
    forward_list<int>::iterator ptr;

    // Inserting value using insert_after()
    // starts insertion from second position
    ptr =  flist.insert_after(flist.begin(), {1, 2, 3});

    // Displaying the forward list
    cout << "The forward list after insert_after operation : ";
    for (int&c : flist)
        cout << c << " ";
    cout << endl;

    // Inserting value using emplace_after()
    // inserts 2 after ptr
    ptr = flist.emplace_after(ptr,2);

    // Displaying the forward list
    cout << "The forward list after emplace_after operation : ";
    for (int&c : flist)
        cout << c << " ";
    cout << endl;

    // Deleting value using erase.after Deleted 2
    // after ptr
    ptr = flist.erase_after(ptr);

    // Displaying the forward list
    cout << "The forward list after erase_after operation : ";
    for (int&c : flist)
        cout << c << " ";
    cout << endl;

    return 0;
}
```

Output:

```
The forward list after insert_after operation : 10 1 2 3 20 30
The forward list after emplace_after operation : 10 1 2 3 2 20 30
The forward list after erase_after operation : 10 1 2 3 2 30
```

7. remove() :- This function removes the particular element from the forward list mentioned in its argument.

8. remove_if() :- This function removes according to the condition in its argument.

```
// C++ code to demonstrate working of remove() and
// remove_if()
#include<iostream>
#include<forward_list>
using namespace std;

int main()
{
    // Initializing forward list
    forward_list<int> flist = {10, 20, 30, 25, 40, 40};

    // Removing element using remove()
    // Removes all occurrences of 40
    flist.remove(40);

    // Displaying the forward list
    cout << "The forward list after remove operation : ";
    for (int&c : flist)
        cout << c << " ";
    cout << endl;

    // Removing according to condition. Removes
    // elements greater than 20. Removes 25 and 30
    flist.remove_if([](int x){ return x>20;});

    // Displaying the forward list
    cout << "The forward list after remove_if operation : ";
    for (int&c : flist)
       cout << c << " ";
    cout << endl;

    return 0;

}
```

Output:

```
The forward list after remove operation : 10 20 30 25
The forward list after remove_if operation : 10 20
```

9. splice_after() :- This function transfers elements from one forward list to other.

```
// C++ code to demonstrate working of
// splice_after()
#include<iostream>
#include<forward_list> // for splice_after()
using namespace std;

int main()
{
    // Initializing forward list
    forward_list<int> flist1 = {10, 20, 30};

    // Initializing second list
    forward_list<int> flist2 = {40, 50, 60};

    // Shifting elements from first to second
    // forward list after 1st position
    flist2.splice_after(flist2.begin(),flist1);

    // Displaying the forward list
    cout << "The forward list after splice_after operation : ";
    for (int&c : flist2)
       cout << c << " ";
    cout << endl;

    return 0;
}
```

Output:

```
The forward list after splice_after operation : 40 10 20 30 50 60
```

Some more methods of forward_list:

- front()– This function is used to reference the first element of the forward list container.
- begin()– begin() function is used to return an iterator pointing to the first element of the forward list container.
- end()– end() function is used to return an iterator pointing to the last element of the list container.
- cbegin()– Returns a constant iterator pointing to the first element of the forward_list.

- cend()– Returns a constant iterator pointing to the past-the-last element of the forward_list.
- before_begin()– Returns a iterator which points to the position before the first element of the forward_list.
- cbefore_begin()– Returns a constant random access iterator which points to the position before the first element of the forward_list.
- max_size()– Returns the maximum number of elements can be held by forward_list.
- resize()– Changes the size of forward_list.

Forward List in C++ | Set 2 (Manipulating Functions)

This article is contributed by **Manjeet Singh** .If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Forward List in C++ | Set 2 (Manipulating Functions)

Virtual Functions and Runtime Polymorphism in C++ | Set 1 (Introduction)

List in C++ | Set 2 (Some Useful Functions)

Forward Iterators in C++

What are Forward declarations in C++

Implementing Forward Iterator in BST

Functions in C/C++

Thread functions in C/C++

Macros vs Functions

Static functions in C

Functions that cannot be overloaded in C++

Inline Functions in C++

Pure Functions

Searching in a map using std::map functions in C++

C | Functions | Question 11

Article Tags : C C++ CPP-Library STL

Practice Tags : STL C CPP

3

1.9

Based on **17** vote(s)

To-do Done

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos

@geeksforgeeks, Some rights reserved