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

Sorting a vector in C++

vector insert() function in C++ STL

string find in C++

map insert() in C++ STL

swap() in C++

map find() function in C++ STL

set find() function in C++ STL

Check if a given graph is Bipartite using DFS

Pre-increment and Post-increment in C/C++

static_cast in C++ | Type Casting operators

map count() function in C++ STL

Sum of array Elements without using loops and recursion

set insert() function in C++ STL

How to return multiple values from a function in C or C++?

vector rbegin() and rend() function in C++ STL

std::any Class in C++

Applications of Pointers in C/C++

Memory leak in C++ and How to avoid it?

map erase() function in C++ STL

STL Priority Queue for Structure or Class

Loader in C/C++

list erase() function in C++ STL

vector emplace() function in C++ STL

Types of Operator Overloading in C++

Check if X can give change to every person in the Queue

set lower_bound() function in C++ STL

Why strcpy and strncpy are not safe to use?

multimap insert() in C++ STL

Modulus function in C++ STL

List in C++ 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.

Below is the program to show the working of some functions of List:

```
#include <iostream>
#include <list>
#include <iterator>
using namespace std;

//function for printing the elements in a list
void showlist(list <int> g)
{
    list <int> :: iterator it;
    for(it = g.begin(); it != g.end(); ++it)
        cout << '\t' << *it;
    cout << '\n';
}

int main()
{

    list <int> gqlist1, gqlist2;


    for (int i = 0; i < 10; ++i)
    {
        gqlist1.push_back(i * 2);
        gqlist2.push_front(i * 3);
    }
    cout << "\nList 1 (gqlist1) is : ";
    showlist(gqlist1);

    cout << "\nList 2 (gqlist2) is : ";
    showlist(gqlist2);

    cout << "\ngqlist1.front() : " << gqlist1.front();
    cout << "\ngqlist1.back() : " << gqlist1.back();

    cout << "\ngqlist1.pop_front() : ";
    gqlist1.pop_front();
    showlist(gqlist1);

    cout << "\ngqlist2.pop_back() : ";
    gqlist2.pop_back();
    showlist(gqlist2);

    cout << "\ngqlist1.reverse() : ";
    gqlist1.reverse();
    showlist(gqlist1);

    cout << "\ngqlist2.sort(): ";
    gqlist2.sort();
    showlist(gqlist2);

    return 0;

}
```

The output of the above program is :

```
List 1 (gqlist1) is :     0    2    4    6
8    10    12    14    16    18

List 2 (gqlist2) is :     27    24    21    18
15    12    9    6    3    0

gqlist1.front() : 0
gqlist1.back() : 18
gqlist1.pop_front() :     2    4    6    8
10    12    14    16    18

gqlist2.pop_back() :     27    24    21    18
15    12    9    6    3

gqlist1.reverse() :     18    16    14    12
10    8    6    4    2

gqlist2.sort():     3    6    9    12
15    18    21    24    27
```

Functions used with List:

- front() – Returns the value of the first element in the list.
- back() – Returns the value of the last element in the list .
- push_front(g) – Adds a new element 'g' at the beginning of the list .
- push_back(g) – Adds a new element 'g' at the end of the list.
- pop_front() – Removes the first element of the list, and reduces size of the list by 1.
- pop_back() – Removes the last element of the list, and reduces size of the list by 1
- list::begin() and list::end() in C++ STL– **begin()** function returns an iterator pointing to the first element of the list
- end()– **end()** function returns an iterator pointing to the theoretical last element which follows the last element.
- list rbegin() and rend() function in C++ STL– **rbegin()** returns a reverse iterator which points to the last element of the list. **rend()** returns a reverse iterator which

points to the position before the beginning of the list.

- list cbegin() and cend() function in C++ STL– **cbegin()** returns a constant random access iterator which points to the beginning of the list. **cend()** returns a constant random access iterator which points to the end of the list.
- list crbegin() and crend() function in C++ STL– **crbegin()** returns a constant reverse iterator which points to the last element of the list i.e reversed beginning of container. **crend()** returns a constant reverse iterator which points to the theoretical element preceding the first element in the list i.e. the reverse end of the list.
- empty() – Returns whether the list is empty(1) or not(0).
- insert() – Inserts new elements in the list before the element at a specified position.
- erase() – Removes a single element or a range of elements from the list.
- assign() – Assigns new elements to list by replacing current elements and resizes the list.
- remove() – Removes all the elements from the list, which are equal to given element.
- list::remove_if() in C++ STL– Used to remove all the values from the list that correspond true to the predicate or condition given as parameter to the function.
- reverse() – Reverses the list.
- size() – Returns the number of elements in the list.
- list resize()function in C++ STL– Used to resize a list container.
- sort() – Sorts the list in increasing order.
- list max_size() function in C++ STL– Returns the maximum number of elements a list container can hold.
- list unique() in C++ STL– Removes all duplicate consecutive elements from the list.
- list::emplace_front() and list::emplace_back() in C++ STL– **emplace_front()** function is used to insert a new element into the list container, the new element is added to the beginning of the list. **emplace_back()** function is used to insert a new element into the list container, the new element is added to the end of the list.
- list::clear() in C++ STL– **clear()** function is used to remove all the elements of the list container, thus making it size 0.
- list::operator= in C++ STL– This operator is used to assign new contents to the container by replacing the existing contents.
- list::swap() in C++ STL– This function is used to swap the contents of one list with another list of same type and size.
- list splice() function in C++ STL– Used to transfer elements from one list to another.

- list merge() function in C++ STL– Merges two sorted lists into one
- list emplace() function in C++ STL– Extends list by inserting new element at a given position.

Recent Articles on C++ list

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

Recommended Posts:

The C++ Standard Template Library (STL)
Set in C++ Standard Template Library (STL)
Map in C++ Standard Template Library (STL)
Multimap in C++ Standard Template Library (STL)
Sort in C++ Standard Template Library (STL)
Deque in C++ Standard Template Library (STL)
Pair in C++ Standard Template Library (STL)
Queue in Standard Template Library (STL)
Multiset in C++ Standard Template Library (STL)
Binary Search in C++ Standard Template Library (STL)
Priority Queue in C++ Standard Template Library (STL)
<iterator> library in C++ STL
<strings> library in C++ STL
<numeric> library in C++ STL
snprintf() in C library

Article Tags : C++ cpp-containers-library cpp-list STL

Practice Tags : STL CPP

4

1.8

To-do Done

Based on **35** vote(s)

Feedback 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 Share this post!