Iterators in C++ STL

Prerequisite : Introduction to Iterators

Iterators are used to point at the memory addresses of STL containers. They are primarily used in sequence of numbers, characters etc. They reduce the complexity and execution time of program.

Operations of iterators :-

1. begin() :- This function is used to return the **beginning position** of the container.

2. end() :- This function is used to return the ***after*** **end position** of the container.

```
// C++ code to demonstrate the working of
// iterator, begin() and end()
#include<iostream>
#include<iterator> // for iterators
#include<vector> // for vectors
using namespace std;
int main()
{
    vector<int> ar = { 1, 2, 3, 4, 5 };

    // Declaring iterator to a vector
    vector<int>::iterator ptr;

    // Displaying vector elements using begin() and end()
    cout << "The vector elements are : ";
    for (ptr = ar.begin(); ptr < ar.end(); ptr++)
        cout << *ptr << " ";

    return 0;
}
```

Output:

```
The vector elements are : 1 2 3 4 5
```

3. advance() :- This function is used to **increment the iterator position** till the specified number mentioned in its arguments.

```
// C++ code to demonstrate the working of
// advance()
#include<iostream>
#include<iterator> // for iterators
#include<vector> // for vectors
using namespace std;
int main()
{
    vector<int> ar = { 1, 2, 3, 4, 5 };

    // Declaring iterator to a vector
    vector<int>::iterator ptr = ar.begin();

    // Using advance() to increment iterator position
    // points to 4
    advance(ptr, 3);

    // Displaying iterator position
    cout << "The position of iterator after advancing is : ";
    cout << *ptr << " ";

    return 0;

}
```

Output:

```
The position of iterator after advancing is : 4
```

4. next() :- This function **returns the new iterator** that the iterator would point after **advancing the positions** mentioned in its arguments.

5. prev() :- This function **returns the new iterator** that the iterator would point **after decrementing the positions** mentioned in its arguments.

```
// C++ code to demonstrate the working of
// next() and prev()
#include<iostream>
#include<iterator> // for iterators
#include<vector> // for vectors
using namespace std;
int main()
{
    vector<int> ar = { 1, 2, 3, 4, 5 };

    // Declaring iterators to a vector
    vector<int>::iterator ptr = ar.begin();
    vector<int>::iterator ftr = ar.end();


    // Using next() to return new iterator
    // points to 4
    auto it = next(ptr, 3);

    // Using prev() to return new iterator
    // points to 3
    auto it1 = prev(ftr, 3);

    // Displaying iterator position
    cout << "The position of new iterator using next() is : ";
    cout << *it << " ";
    cout << endl;

    // Displaying iterator position
    cout << "The position of new iterator using prev()  is : ";
    cout << *it1 << " ";
    cout << endl;

    return 0;
}
```

Output:

```
The position of new iterator using next() is : 4
The position of new iterator using prev()  is : 3
```

6. inserter() :- This function is used to **insert the elements at any position** in the container. It accepts **2 arguments, the container and iterator to position where the elements have to be inserted**.

```cpp
// C++ code to demonstrate the working of
// inserter()
#include<iostream>
#include<iterator> // for iterators
#include<vector> // for vectors
using namespace std;
int main()
{
    vector<int> ar = { 1, 2, 3, 4, 5 };
    vector<int> ar1 = {10, 20, 30};

    // Declaring iterator to a vector
    vector<int>::iterator ptr = ar.begin();

    // Using advance to set position
    advance(ptr, 3);

    // copying 1 vector elements in other using inserter()
    // inserts ar1 after 3rd position in ar
    copy(ar1.begin(), ar1.end(), inserter(ar,ptr));

    // Displaying new vector elements
    cout << "The new vector after inserting elements is : ";
    for (int &x : ar)
        cout << x << " ";

    return 0;
}
```

Output:

```
The new vector after inserting elements is : 1 2 3 10 20 30 4 5
```

Types of Iterators :

1. Input Iterators
2. Output Iterators
3. Forward Iterator
4. Bidirectional Iterators
5. Random-Access Iterators

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:

Bidirectional Iterators in C++

Output Iterators in C++

Forward Iterators in C++

Input Iterators in C++

Introduction to Iterators in C++

Random-access Iterators in C++

Conditional or Ternary Operator (?:) in C/C++

forward_list insert_after() function in C++ STL

Count of distinct remainders when N is divided by all the numbers from the range [1, N]

C++ | asm declaration

Pointers and References in C++

Strings in C++ and How to Create them?

Enum Classes in C++ and Their Advantage over Enum DataType

Deque vs Vector in C++ STL

Improved By : IkamaruEnauki

Article Tags : C++ cpp-iterator CPP-Library STL

Practice Tags : STL CPP

16

2.6

Based on **25** 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