Iterators in C++ STL

Iterators are used to point at the memory addresses of STL containers. They are primarily used in a sequence of numbers, characters etc. We can use iterators to move through the contents of the container. They can be visualized as something similar to a pointer pointing to some location and we can access content at that particular location using them.

Basic Operations of iterators :-

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

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

- **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;

}
```

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

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

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

- **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**.

```
// 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;
}
```

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