Vector in C++ STL

Difficulty Level : Easy • Last Updated : 04 Sep, 2019

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

Certain functions associated with the vector are:

Iterators

1. begin() – Returns an iterator pointing to the first element in the vector

2. end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector
3. rbegin() – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
4. rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)
5. cbegin() – Returns a constant iterator pointing to the first element in the vector.
6. cend() – Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.
7. crbegin() – Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
8. crend() – Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

```
// C++ program to illustra
// iterators in vector
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> g1;

    for (int i = 1; i <= 5
        g1.push_back(i);

    cout << "Output of beg
    for (auto i = g1.begin
        cout << *i << " ";

    cout << "\nOutput of c
    for (auto i = g1.cbegi
        cout << *i << " ";

    cout << "\nOutput of r
    for (auto ir = g1.rbeg
        cout << *ir << " "

    cout << "\nOutput of c
    for (auto ir = g1.crbe
        cout << *ir << " "

    return 0;
}
```

Output:

```
Output of begin and end: 1
Output of cbegin and cend:
Output of rbegin and rend:
Output of crbegin and crend
```

Capacity

1. size() – Returns the number of elements in the vector.
2. max_size() – Returns the maximum number of elements that the vector can hold.
3. capacity() – Returns the size of the storage space currently allocated to the vector expressed as number of elements.
4. resize(n) – Resizes the container so that it contains 'n' elements.
5. empty() – Returns whether the container is empty.
6. shrink_to_fit() – Reduces the capacity of the container to fit its size and destroys all elements beyond the capacity.
7. reserve() – Requests that the vector capacity be at least enough to contain n elements.

```
// C++ program to illustra
// capacity function in ve
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> g1;

    for (int i = 1; i <= 5
        g1.push_back(i);

    cout << "Size : " << g
    cout << "\nCapacity :
    cout << "\nMax_Size :

    // resizes the vector
    g1.resize(4);

    // prints the vector s
    cout << "\nSize : " <<

    // checks if the vecto
    if (g1.empty() == fals
        cout << "\nVector
    else
        cout << "\nVector

    // Shrinks the vector
    g1.shrink_to_fit();
    cout << "\nVector elem
    for (auto it = g1.begi
        cout << *it << " "

    return 0;
}
```

Output:

```
Size : 5
Capacity : 8
Max_Size : 4611686018427387
Size : 4
Vector is not empty
Vector elements are: 1 2 3
```

Element access:

1. reference operator [g] – Returns a reference to the element at position 'g' in the vector
2. at(g) – Returns a reference to the element at position 'g' in the vector
3. front() – Returns a reference to the first element in the vector
4. back() – Returns a reference to the last element in the vector
5. data() – Returns a direct pointer to the memory array used internally by the vector to store its owned elements.

```
// C++ program to illustra
// element accesser in vec
#include <bits/stdc++.h>
using namespace std;

int main()
{
    vector<int> g1;

    for (int i = 1; i <= 1
        g1.push_back(i * 1

    cout << "\nReference o

    cout << "\nat : g1.at(

    cout << "\nfront() : g

    cout << "\nback() : g1

    // pointer to the firs
    int* pos = g1.data();

    cout << "\nThe first e
    return 0;
}
```

Output:

```
Reference operator [g] : g1
at : g1.at(4) = 50
front() : g1.front() = 10
back() : g1.back() = 100
The first element is 10
```

Modifiers:

1. assign() – It assigns new value to the vector elements by replacing old ones
2. push_back() – It push the elements into a vector from the back
3. pop_back() – It is used to pop or

specified position
5. erase() – It is used to remove elements from a container from the specified position or range.
6. swap() – It is used to swap the contents of one vector with another vector of same type. Sizes may differ.
7. clear() – It is used to remove all the elements of the vector container
8. emplace() – It extends the container by inserting new element at position
9. emplace_back() – It is used to insert a new element into the vector container, the new element is added to the end of the vector

.

```cpp
// C++ program to illustra
// Modifiers in vector
#include <bits/stdc++.h>
#include <vector>
using namespace std;

int main()
{
    // Assign vector
    vector<int> v;

    // fill the array with
    v.assign(5, 10);

    cout << "The vector el
    for (int i = 0; i < v.
        cout << v[i] << "

    // inserts 15 to the l
    v.push_back(15);
    int n = v.size();
    cout << "\nThe last el

    // removes last elemen
    v.pop_back();

    // prints the vector
    cout << "\nThe vector
    for (int i = 0; i < v.
        cout << v[i] << "

    // inserts 5 at the be
    v.insert(v.begin(), 5)

    cout << "\nThe first e

    // removes the first e
    v.erase(v.begin());

    cout << "\nThe first e

    // inserts at the begi
    v.emplace(v.begin(), 5
    cout << "\nThe first e

    // Inserts 20 at the e
    v.emplace_back(20);
    n = v.size();
    cout << "\nThe last el

    // erases the vector
    v.clear();
    cout << "\nVector size

    // two vector to perfo
```

```
        vector<int> v1, v2;
        v1.push_back(1);
        v1.push_back(2);
        v2.push_back(3);
        v2.push_back(4);

        cout << "\n\nVector 1:
        for (int i = 0; i < v1
            cout << v1[i] << "

        cout << "\nVector 2: "
        for (int i = 0; i < v2
            cout << v2[i] << "

        // Swaps v1 and v2
        v1.swap(v2);

        cout << "\nAfter Swap
        for (int i = 0; i < v1
            cout << v1[i] << "

        cout << "\nVector 2: "
        for (int i = 0; i < v2
            cout << v2[i] << "
    }
```

Output:

```
The vector elements are: 10
The last element is: 15
The vector elements are: 10
The first element is: 5
The first element is: 10
The first element is: 5
The last element is: 20
Vector size after erase():

Vector 1: 1 2
Vector 2: 3 4
After Swap
Vector 1: 3 4
Vector 2: 1 2
```

All Vector Functions :

Please write comments if you find

- vector::begin() and vector::end()
- vector rbegin() and rend()
- vector::cbegin() and vector::cend()
- vector::crend() and vector::crbegin()
- vector::assign()
- vector::at()
- vector::back()
- vector::capacity()
- vector::clear()
- vector::push_back()
- vector::pop_back()
- vector::empty()
- vector::erase()
- vector::size()
- vector::swap()
- vector::reserve()
- vector::resize()
- vector::shrink_to_fit()
- vector::operator=
- vector::operator[]
- vector::front()
- vector::data()
- vector::emplace_back()
- vector::emplace()
- vector::max_size()
- vector::insert()

anything incorrect, or you want to share more information about the topic discussed above.

Rated as one of the most sought after skills in the industry, own the basics of coding with our **C++ STL** Course and master the very concepts by intense problem-solving.

RECOMMENDED ARTICLES

Page : 1 2 3