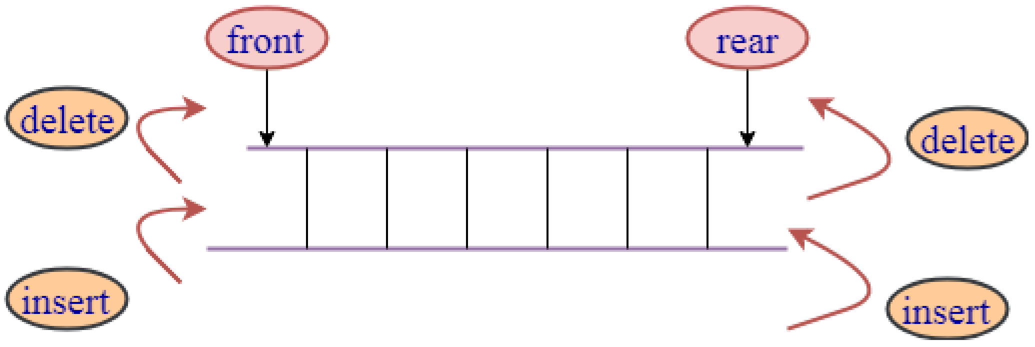
Deque

Deque stands for double ended queue. It generalizes the queue data structure i.e insertion and deletion can be performed from both the ends either front or back.

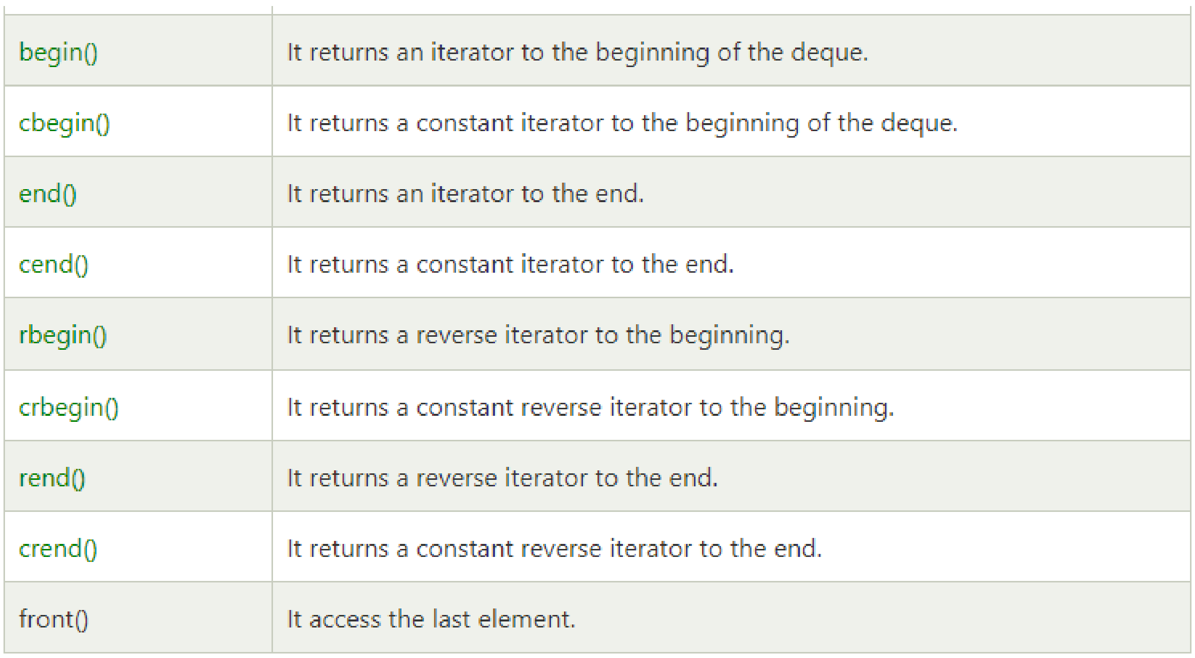


Syntax for creating a deque object:

deque<object_type> deque_name;

C++ Deque Functions

Method	Description
assign()	It assigns new content and replacing the old one.
emplace()	It adds a new element at a specified position.
emplace_back()	It adds a new element at the end.
emplace_front()	It adds a new element in the beginning of a deque.
insert()	It adds a new element just before the specified position.
push_back()	It adds a new element at the end of the container.
push_front()	It adds a new element at the beginning of the container.
pop_back()	It deletes the last element from the deque.
pop_front()	It deletes the first element from the deque.
swap()	It exchanges the contents of two deques.
clear()	It removes all the contents of the deque.
empty()	It checks whether the container is empty or not.
erase()	It removes the elements.
max_size()	It determines the maximum size of the deque.
resize()	It changes the size of the deque.
shrink_to_fit()	It reduces the memory to fit the size of the deque.
size()	It returns the number of elements.
at()	It access the element at position pos.
operator[]()	It access the element at position pos.
operator=()	It assigns new contents to the container.
back()	It access the last element.



begin()	It returns an iterator to the beginning of the deque.
cbegin()	It returns a constant iterator to the beginning of the deque.
end()	It returns an iterator to the end.
cend()	It returns a constant iterator to the end.
rbegin()	It returns a reverse iterator to the beginning.
crbegin()	It returns a constant reverse iterator to the beginning.
rend()	It returns a reverse iterator to the end.
crend()	It returns a constant reverse iterator to the end.
front()	It access the last element.

Example: CPP Program to implement Deque in STL

```
#include <deque>
#include <iostream>

using namespace std;

void showdq(deque<int> g)
{
        deque<int>::iterator it;
        for (it = g.begin(); it != g.end(); ++it)
                cout << '\t' << *it;
        cout << '\n';
}

int main()
{
        deque<int> gquiz;
        gquiz.push_back(10);
        gquiz.push_front(20);
        gquiz.push_back(30);
        gquiz.push_front(15);
        cout << "The deque gquiz is : ";
```

```
    showdq(gquiz);

    cout << "\ngquiz.size() : " << gquiz.size();
    cout << "\ngquiz.max_size() : " << gquiz.max_size();

    cout << "\ngquiz.at(2) : " << gquiz.at(2);
    cout << "\ngquiz.front() : " << gquiz.front();
    cout << "\ngquiz.back() : " << gquiz.back();

    cout << "\ngquiz.pop_front() : ";
    gquiz.pop_front();
    showdq(gquiz);

    cout << "\ngquiz.pop_back() : ";
    gquiz.pop_back();
    showdq(gquiz);

    return 0;
}
```