BAKI ALİ NEFT MƏKTƏBİ
BAKU HIGHER OIL SCHOOL

Programming and Computer Applications-2

STL queue and deque Containers

Instructor : PhD, Associate Professor Leyla Muradkhanli

The STL `queue` and `deque` Containers

- **`queue`:** container ADT that can be used to provide queue as a **`vector`**, **`list`**, or **`deque`**. Has member functions to **`enqueue`** (**`push`**) and **`dequeue`** (**`pop`**)
- **`deque`:** a double-ended queue. Has member functions to **`enqueue`** (**`push_back`**) and **`dequeue`** (**`pop_front`**)

Defining a `queue`

- Defining a queue of **`chars`**, named cQueue, implemented using a **`deque`**:

```
deque<char> cQueue;
```

- implemented using a **`queue`**:

```
queue<char> cQueue;
```

- implemented using a **`list`**:

```
queue< char, list<char> > cQueue;
```

- Spaces are required between consecutive >>, << symbols

The STL queue Container

The functions supported by queue are :

- **empty()** – Returns whether the queue is empty
- **size()** – Returns the size of the queue
- **front()** – Returns a reference to the first element of the queue
- **back()** – Returns a reference to the last element of the queue
- **push(e)** – Adds the element ‘e’ at the end of the queue
- **pop()** – Deletes the first element of the queue

The STL queue Container

```
#include <iostream>
#include <queue>
using namespace std;
void display(queue <int> q1)
{
 queue <int> q = q1;
 while (!q.empty())
 {
 cout << '\t' << q.front();
 q.pop();
 }
 cout << '\n';
}
```

The STL queue Container

```
int main()
{
 queue <int> myqueue;
 myqueue.push(10);
 myqueue.push(20);
 cout << "The myqueue is : ";
 display(myqueue);
 cout << "\nmyqueue.size() : " << myqueue.size();
 cout << "\nmyqueue.front() : " << myqueue.front();
 cout << "\nmyqueue.back() : " << myqueue.back();
 myqueue.pop();
 cout<<"\nAfter pop myqueue is : ";
 display(myqueue);
return 0;
}
```

The STL queue Container

The output of the above program is :

```
The myqueue is : 10        20
myqueue.size() : 2
myqueue.front() : 10
myqueue.back() : 20
After pop myqueue is : 20
```

The STL deque Container

Double ended queues are sequence containers with the feature of expansion and contraction on both the ends. They are similar to vectors, but are more efficient in case of insertion and deletion of elements at the end, and also the beginning. Unlike vectors, contiguous storage allocation may not be guaranteed.

The functions for deque are same as vector, with an addition of push and pop operations for both front and back.

The STL deque Container

```
#include <iostream>
#include <deque>
using namespace std;
void display(deque <int> q)
{
 for (auto it = q.begin(); it != q.end(); ++it)
 cout << '\t' << *it;
 cout << '\n';
}
```

The STL deque Container

```
int main()
{
 deque <int> q1;
 q1.push_back(10);
 q1.push_front(20);
 q1.push_back(30);
 q1.push_front(50);
 cout << "The deque q1 is : ";
 display(q1);
```

The STL deque Container

```
 cout << "\nq1.size() : " << q1.size();
 cout << "\nq1.max_size() : " << q1.max_size();
 cout << "\nq1.at(2) : " << q1.at(2);
 cout << "\nq1.front() : " << q1.front();
 cout << "\nq1.back() : " << q1.back();
 q1.pop_front();
 cout << "\nAfter pop_front() queue is : ";
 display(q1);
 q1.pop_back();
 cout << "\nAfter pop_back() queue is : ";
 display(q1);
 return 0;
}
```

The STL deque Container

```
The output of the above program is :
The deque q1 is : 50  20   10   30
q1.size() : 4
q1.max_size() : 1073741823
q1.at(2) : 10
q1.front() : 50
q1.back() : 30
After pop_front() queue is : 20 10 30
After pop_back() queue is : 20 10
```