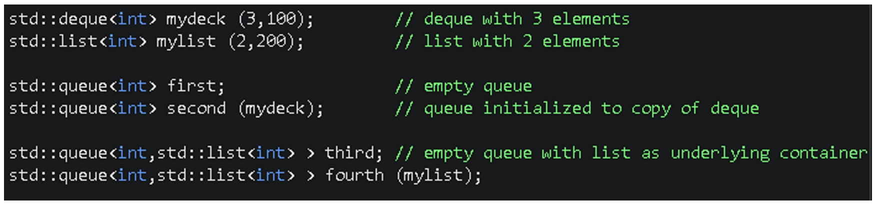
Stack-Queue-PriorityQueue

09 June 2021 07:07

1. Queue - container adaptor, FIFO -> deque(default), list
   https://www.cplusplus.com/reference/queue/queue
   front, back, push, pop, emplace, size



```
std::deque<int> mydeck (3,100);        // deque with 3 elements
std::list<int> mylist (2,200);         // list with 2 elements

std::queue<int> first;                 // empty queue
std::queue<int> second (mydeck);       // queue initialized to copy of deque

std::queue<int,std::list<int> > third; // empty queue with list as underlying container
std::queue<int,std::list<int> > fourth (mylist);
```

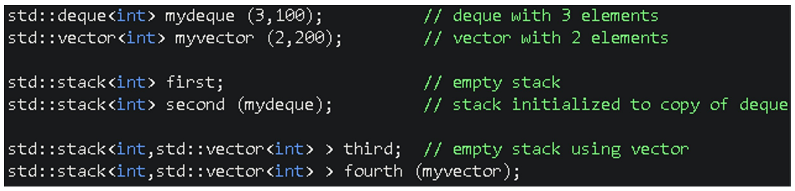
2. Priority Queue - container adaptor -> vector(default).. Deque
   https://www.cplusplus.com/reference/queue/priority_queue/
   top, push, pop, emplace, size, empty

   Uses algo functions such as push_heap, pop_heap
   1. push -> push_back and make_heap/push_heap (push and make heap)
   2. Pop -> make_heap/pop_heap and pop_back

```
class mycomparison
{
  bool reverse;
public:
  mycomparison(const bool& revparam=false){reverse=revparam;}
  bool operator() (const int& lhs, const int&rhs) const
  {
    if (reverse) return (lhs>rhs);
    else return (lhs<rhs);
  }
};

int main ()
{
  int myints[]= {10,60,50,20};
  std::priority_queue<int> first;
  std::priority_queue<int> second (myints, myints+4);
  std::priority_queue<int, std::vector<int>, std::greater<int>> third (myints, myints+4);
  std::priority_queue<int, std::vector<int>, mycomparison> fourth;                    // less-than comparison
  std::priority_queue<int, std::vector<int>, mycomparison> fifth (mycomparison(true));   // greater-than comparison
}
```

3. Stack - container adaptor, LIFO -> deque(default) … vector, list
   top, push, pop, emplace, size, empty
   https://www.cplusplus.com/reference/stack/stack/



```
std::deque<int> mydeque (3,100);          // deque with 3 elements
std::vector<int> myvector (2,200);        // vector with 2 elements

std::stack<int> first;                    // empty stack
std::stack<int> second (mydeque);         // stack initialized to copy of deque

std::stack<int,std::vector<int> > third;  // empty stack using vector
std::stack<int,std::vector<int> > fourth (myvector);
```

4. F
   I
5. F
   I
6. F
7. F
   I
8. F
   I
9. F

10. F
11. F
12. F
13. D
14. F
15. F
16. F
17. F
18. F
19. F
20. F
21. F
22. F
23. F
24. D
25. F
26. F
27. F
28. F
29. F
30. F
31. F
32. F
33. F
34. F
35. D
36. F
37. F
38. F
39. F
40. F

I

41. F

    I

42. F

    I

43. F

    I

44. F

    I