

Queue

A queue follows the opposite principle of stack. It is a very powerful data structure in its own right.



The `std::queue` follows the FIFO principle (First In First Out). The queue `que`, which needs the header `<queue>`, has four special methods.

With `que.push(e)` you can insert an element `e` at the end of the queue and remove the first element from the queue with `que.pop()`. `que.back()` enables you to refer to the last element in the `que`, `que.front()` to the first element in the `que`. `std::queue` has similar characteristics as `std::stack`. So you can compare `std::queue` instances and get their sizes. The operations of the queue have constant complexity.

```
#include <iostream>
#include <queue>

int main(){
    std::queue<int> myQueue;

    std::cout << myQueue.empty() << std::endl;    // true
    std::cout << myQueue.size() << std::endl;    // 0

    myQueue.push(1);
    myQueue.push(2);
    myQueue.push(3);
    std::cout << myQueue.back() << std::endl;    // 3
    std::cout << myQueue.front() << std::endl;    // 1

    while (!myQueue.empty()){
        std::cout << myQueue.back() << " ";
        std::cout << myQueue.front() << " : ";
        myQueue.pop();
    }
    // 3 1 : 3 2 : 3 3
```

```
std::cout << std::endl << myQueue.empty() << std::endl;    //1 (denotes true)
std::cout << myQueue.size() << std::endl;    // 0

return 0;
}
```



std::queue