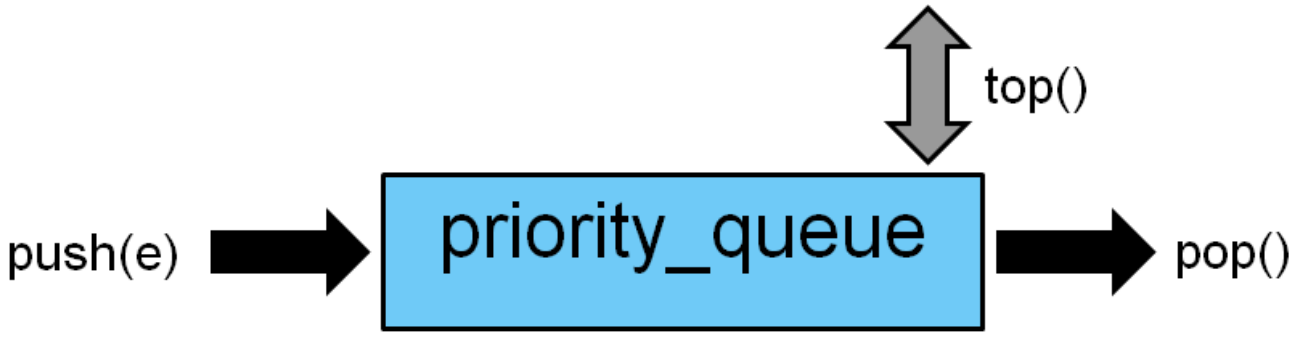
Priority Queue

By combining a queue and order, we get priority queues!



The std::priority_queue is a reduced `std::queue`. It needs the header `<queue>`.

The difference to the `std::queue` is, that their biggest element is always at the top of the priority queue. `std::priority_queue pri` uses by default the comparison operator `std::less`. Similar to `std::queue`, `pri.push(e)` inserts a new element `e` into the priority queue. `pri.pop()` removes the first element of the `pri`, but does that with logarithmic complexity. With `pri.top()` you can reference the first element in the priority queue, which is the greatest one. The `std::priority_queue` knows its size, but didn't support the comparison operator on their instances.

```
// priorityQueue.cpp
#include <iostream>
#include <queue>

int main(){
  std::priority_queue<int> myPriorityQueue;

  std::cout << "is empty:\t" << myPriorityQueue.empty() << std::endl;   // 1 (denotes true)
  std::cout << "size:\t\t" << myPriorityQueue.size() << std::endl;    // 0

  myPriorityQueue.push(3);
  myPriorityQueue.push(1);
  myPriorityQueue.push(2);
  std::cout << "top:\t\t" << myPriorityQueue.top() << std::endl;     // 3

  std::cout<< "Data:\t";
  while (!myPriorityQueue.empty()){
    std::cout << myPriorityQueue.top() << " ";
```

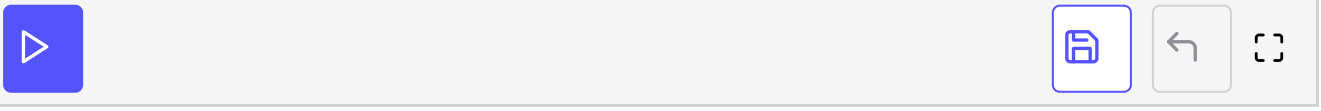
```
    myPriorityQueue.pop();
  }                                                    // 3 2 1
  std::cout << std::endl;


  std::cout << "is empty:\t" << myPriorityQueue.empty() << std::endl;   // 1 (denotes true)
  std::cout << "size:\t\t" << myPriorityQueue.size() << std::endl;    // 0

  std::priority_queue<std::string, std::vector<std::string>,
                      std::greater<std::string>> myPriorityQueue2;

  myPriorityQueue2.push("Only");
  myPriorityQueue2.push("for");
  myPriorityQueue2.push("testing");
  myPriorityQueue2.push("purpose");
  myPriorityQueue2.push(".");

  while (!myPriorityQueue2.empty()){
    std::cout << myPriorityQueue2.top() << " ";
    myPriorityQueue2.pop();
  }                                  // . Only for purpose testing
  return 0;
}
```



std::priority_queue