# Priority queue

A **C++ priority queue** is a type of container adapter, specifically designed such that the first element of the queue is either the greatest or the smallest of all elements in the queue, and elements are in non-increasing or non-decreasing order (hence we can see that each element of the queue has a priority (fixed order).

The top element is always the greatest by default. We can also change it to the smallest element at the top. Priority queues are built on the top of the max heap and use an array or vector as an internal structure. In simple terms, **STL Priority Queue** is the implementation of Heap Data Structure.

1. To use priority queue, you have to include the 'queue' header file

```
#include <queue>
```

2. The syntax to declare a priority queue is:

```
std::priority_queue<data_type> name;
```

3. The syntax to declare a minimum heap (minimum priority queue) is:

```
std::priority_queue<int, std::vector<int>, std::greater<int>> name;
```

where,

- 'int' is the type of elements you want to store in the priority queue. In this case, it's an integer. You can replace **int** with any other data type you need.
- 'vector int' is the type of internal container used to store these elements. **std::priority_queue** is not a container in itself but a container adopter. It wraps other containers. In this example, we're using a **vector**, but you could choose a different container that supports front(), push_back(), and pop_back() methods.
- 'greater int' is a custom comparison function. This determines how the elements are ordered within the priority queue. In this specific example, greater int sets up a **min-heap**. It means that the smallest element will be at the top of the queue.

## Functions

All the functions work similar to queue except that the elements are arranged according to the priority.

- **push()** - Time complexity of log(n)
- **top()** - Time complexity of O(1)
- **pop()** - Time complexity of log(n)