# Priority Queue

The priority queue in C++ is a derived container in STL that considers only the highest priority element. The queue follows the FIFO policy while priority queue pops the elements based on the priority, i.e., the highest priority element is popped first.

It is similar to the ordinary queue in certain aspects but differs in the following ways:

- In a priority queue, every element in the queue is associated with some priority, but priority does not exist in a queue data structure.
- The element with the highest priority in a priority queue will be removed first while queue follows the FIFO(First-In-First-Out) policy means the element which is inserted first will be deleted first.
- If more than one element exists with the same priority, then the order of the element in a queue will be taken into consideration.

**Note:** The priority queue is the extended version of a normal queue except that the element with the highest priority will be removed first from the priority queue.

**Syntax of Priority Queue**

priority_queue<int> variable_name;

## Member Function of Priority Queue

| Function | Description |
|---|---|
| **push()** | It inserts a new element in a priority queue. |
| **pop()** | It removes the top element from the queue, which has the highest priority. |
| **top()** | This function is used to address the topmost element of a priority queue. |
| **size()** | It determines the size of a priority queue. |
| **empty()** | It verifies whether the queue is empty or not. Based on the verification, it returns the status. |
| **swap()** | It swaps the elements of a priority queue with another queue having the same type and size. |
| **emplace()** | It inserts a new element at the top of the priority queue. |

## Example:

```
#include <iostream>
#include<queue>
using namespace std;
int main()
{
 priority_queue<int> p;  // variable declaration.
 p.push(10); // inserting 10 in a queue, top=10
 p.push(30); // inserting 30 in a queue, top=30
 p.push(20); // inserting 20 in a queue, top=20
 cout<<"Number of elements available in 'p' :"<<p.size()<<endl;
 while(!p.empty())
 {
    std::cout << p.top() << std::endl;
    p.pop();
 }
 return 0;
}
```

In the above code, we have created a priority queue in which we insert three elements, i.e., 10, 30, 20. After inserting the elements, we display all the elements of a priority queue by using a while loop.

Output
Number of elements available in 'p' :3
30
20
10

**Example2:**

```
#include <iostream>
#include<queue>
using namespace std;
int main()
{
  priority_queue<int> p;  // priority queue declaration
  priority_queue<int> q;  // priority queue declaration
  p.push(1); // inserting element '1' in p.
  p.push(2); // inserting element '2' in p.
  p.push(3); // inserting element '3' in p.
  p.push(4); // inserting element '4' in p.
  q.push(5); // inserting element '5' in q.
  q.push(6); // inserting element '6' in q.
  q.push(7); // inserting element '7' in q.
  q.push(8); // inserting element '8' in q.
  p.swap(q);
  std::cout << "Elements of p are : " << std::endl;
  while(!p.empty())
  {
    std::cout << p.top() << std::endl;
     p.pop();
  }
  std::cout << "Elements of q are :" << std::endl;
   while(!q.empty())
  {
    std::cout << q.top() << std::endl;
```

```
    q.pop();
  }
   return 0;
}
```

In the above code, we have declared two priority queues, i.e., p and q. We inserted four elements in 'p' priority queue and four in 'q' priority queue. After inserting the elements, we swap the elements of 'p' queue with 'q' queue by using a swap() function.

Output

```
Elements of p are :
8
7
6
5
Elements of q are :
4
3
2
1
```