**Priority Queue in C++ Standard Template Library (STL):**

Priority queues are a type of **container** adapters, specifically designed such that the **first** element of the queue is the greatest of all elements in the queue and elements are in non increasing order(hence we can see that each element of the queue has a priority{fixed order}).

// Note that by default C++ creates a **max-heap** for priority queue

#include <iostream>

#include <queue>

using namespace std;

void showpq(**priority\_queue** < int > **q**)

{

**priority\_queue** < int > **g** = q;

while ( !**g**.**empty()** )

{

cout << **g**. **top()**;

**g**.**pop();**

}

cout << '\n';

}

int main ()

{

**priority\_queue** < int > **pq**;

pq.**push**(10);

pq.**push**(30);

pq.**push**(20);

pq.**push**(5);

pq.**push**(1);

cout << "The priority queue pq is : ";

showpq( **pq** );

cout << " pq.size() : " << pq .**size()**;

cout << "pq .top() : " << pq .**top() ;**

cout << "pq.pop() : ";

pq .**pop();**

showpq( **pq** );

return 0;

}

**Output:**

The priority queue pq is : 30 20 10 5 1

pq.size() : 5

pq.top() : 30

pq.pop() : 20 10 5 1

**How to create a min heap for priority queue?**  
C++ syntax for that :

***priority\_queue <int, vector<int>, greater<int>> g ;***

// C++ program to demonstrate **min heap**

#include <iostream>

#include <queue>

using namespace std;

void showpq( **priority\_queue** **< int**, **vector<int>**, **greater<int>** **>** **q**)

{

**priority\_queue** **< int**, **vector<int>**, **greater<int>** **>** **g** = **q**;

while ( !**g**.**empty()** )

{ cout<< g.**top()**;

g.**pop()**;

}

}

int main ()

{

**priority\_queue** **< int**, **vector<int>**, **greater<int>** **>** **pq**;

pq.**push**(10);

pq.**push**(30);

pq.**push**(20);

pq.**push**(5);

pq.**push**(1);

cout << "The priority queue pq is : ";

showpq( **pq** );

cout << " pq.size() : " << pq .**size()**;

cout << "pq .top() : " << pq .**top() ;**

cout << "pq.pop() : ";

pq.**pop();**

showpq( **pq** );

return 0;

}

**Output:**

The priority queue pq is : 1 5 10 20 30

pq.size() : 5

pq.top() : 1

pq.pop() : 5 10 20 30

**Note :** The above syntax is difficult to remembers, so in case of numeric values, we can multiply values with ( -1) and use **max heap** to get the effect of **min heap**

# How to implement Min Heap using STL?

In C++ STL, there is [priority\_queue](https://www.geeksforgeeks.org/priority-queue-container-adaptors-the-c-standard-template-library-stl/) that can directly be used to implement **Max Heap**

// C++ program to show that priority\_queue is by default a **Max Heap**

#include <bits/stdc++.h>

using namespace std;

int main ()

{

// Creates a **max heap**

**priority\_queue** < int > **pq**;

pq.**push**(5);

pq.**push**(1);

pq.**push**(10);

pq.**push**(30);

pq.**push**(20);

// One by one extract items from **max heap**

while ( **pq**.**empty()** == **false**)

{

cout << **pq**.**top()** << " ";

**pq**.**pop()**;

}

return 0;

}

Output :

30 20 10 5 1

Since elements are printed in descending order, we have a max heap by default.

**How to implement Min Heap?**  
**priority\_queue** supports a constructor that requires two extra arguments to make a **min heap**.

priority\_queue <Type, vector<Type>, ComparisonType > min\_heap;

// C++ program to use priority\_queue to implement **min heap**

#include <bits/stdc++.h>

using namespace std;

int main ()

{

**priority\_queue** **< int**, **vector<int>**, **greater<int>** **>** **pq**;

pq.**push**(5);

pq.**push**(1);

pq.**push**(10);

pq.**push**(30);

pq.**push**(20);

// One by one extract items from min heap

while ( **pq**.**empty()** == **false**)

{

cout << **pq**.**top()** << " ";

**pq**.**pop()**;

}

return 0;

}

**Output :**

1 5 10 20 30

**How to make a min heap of user defined class?**  
Let us consider below example where we build a **min heap** of 2 D points ordered by X axis.

// C++ program to use priority\_queue to implement Min Heap

// for user defined class

#include <bits/stdc++.h>

using namespace std;

// User defined class, Point

class **Point**

{

int x;

int y;

public:

Point(int \_x, int \_y)

{

x = \_x;

y = \_y;

}

int getX() const { return x; }

int getY() const { return y; }

};

// To compare two points

class **myComparator**

{

public:

int **operator()** (const Point& p1, const Point& p2)

{

return p1.getX() **>** p2.getX();

}

};

int main ()

{

// Creates a **Min heap** of points (order by x coordinate)

**priority\_queue** **< Point**, **vector<Point>**, **myComparator** **>** **pq**;

// Insert points into the **min heap**

pq.**push(** **Point**(10, 2) **)**;

pq.**push(Point** (2, 1) **)**;

pq.**push(Point** (1, 5) **)**;

// One by one extract items from min heap

while ( **pq**.**empty()** == **false**)

{

Point p = pq.**top();**

cout << "(" << p.getX() << ", " << p.getY() << ")";

**pq**.**pop()**;

}

return 0;

}

**Output :**

(1, 5)

(2, 1)

(10, 2)

**Methods of priority queue are:**

* [priority\_queue::empty() in C++ STL](https://www.geeksforgeeks.org/priority_queueempty-priority_queuesize-c-stl/)– **empty()** function returns whether the queue is empty.
* [priority\_queue::size() in C++ STL](https://www.geeksforgeeks.org/priority_queueempty-priority_queuesize-c-stl/)– **size()** function returns the size of the queue.
* [priority\_queue::top() in C++ STL](https://www.geeksforgeeks.org/priority_queuetop-c-stl/)– Returns a reference to the top most element of the queue
* [priority\_queue::push() in C++ STL](https://www.geeksforgeeks.org/priority_queuepush-priority_queuepop-c-stl/)**– push(g)** function adds the element ‘g’ at the end of the queue.
* [priority\_queue::pop() in C++ STL](https://www.geeksforgeeks.org/priority_queuepush-priority_queuepop-c-stl/)– **pop()** function deletes the first element of the queue.
* [priority\_queue::swap() in C++ STL](https://www.geeksforgeeks.org/priority_queueswap-c-stl/)– This function is used to swap the contents of one priority queue with another priority queue of same type and size.
* [priority\_queue::emplace() in C++ STL](https://www.geeksforgeeks.org/priority_queueemplace-c-stl/)– This function is used to insert a new element into the priority queue container, the new element is added to the top of the priority queue.
* [priority\_queue value\_type in C++ STL](https://www.geeksforgeeks.org/priority_queue-value_type-in-c-stl/)– Represents the type of object stored as an element in a priority\_queue. It acts as a synonym for the template parameter.