Priority Queue using array in C++

Priority Queue is an extension of the Queue data structure where each element has a particular priority associated with it. It is based on the priority value, the elements from the queue are deleted.

Operations on Priority Queue:

- **enqueue():** This function is used to insert new data into the queue.
- **dequeue():** This function removes the element with the highest priority from the queue.
- **peek()/top():** This function is used to get the highest priority element in the queue without removing it from the queue.

Approach: The idea is to create a structure to store the value and priority of the element and then create an array of that structure to store elements. Below are the functionalities that are to be implemented:

- **enqueue():** It is used to insert the element at the end of the queue.
- **peek():**
  - Traverse across the priority queue and find the element with the highest priority and return its index.
  - In the case of multiple elements with the same priority, find the element with the highest value having the highest priority.
- **dequeue():**
  - Find the index with the highest priority using the **peek()** function let's call that position as **ind**, and then shift the position of all the elements after the position **ind** one position to the left.
  - Decrease the size by one.

Below is the implementation of the above approach:

- C++

```
// C++ program for the above approach

#include <bits/stdc++.h>
using namespace std;

// Structure for the elements in the
// priority queue
struct item {
    int value;
    int priority;
};

// Store the element of a priority queue
item pr[100000];
```

```
// Pointer to the last index
int size = -1;

// Function to insert a new element
// into priority queue
void enqueue(int value, int priority)
{
    // Increase the size
    size++;

    // Insert the element
    pr[size].value = value;
    pr[size].priority = priority;
}

// Function to check the top element
int peek()
{
    int highestPriority = INT_MIN;
    int ind = -1;

    // Check for the element with
    // highest priority
    for (int i = 0; i <= size; i++) {

        // If priority is same choose
        // the element with the
        // highest value
        if (highestPriority
                == pr[i].priority
            && ind > -1
            && pr[ind].value
                   < pr[i].value) {
            highestPriority = pr[i].priority;
            ind = i;
        }
        else if (highestPriority
                 < pr[i].priority) {
            highestPriority = pr[i].priority;
            ind = i;
        }
    }

    // Return position of the element
    return ind;
```

```
}

// Function to remove the element with
// the highest priority
void dequeue()
{
    // Find the position of the element
    // with highest priority
    int ind = peek();

    // Shift the element one index before
    // from the position of the element
    // with highest priority is found
    for (int i = ind; i < size; i++) {
        pr[i] = pr[i + 1];
    }

    // Decrease the size of the
    // priority queue by one
    size--;
}

// Driver Code
int main()
{
    // Function Call to insert elements
    // as per the priority
    enqueue(10, 2);
    enqueue(14, 4);
    enqueue(16, 4);
    enqueue(12, 3);

    // Stores the top element
    // at the moment
    int ind = peek();

    cout << pr[ind].value << endl;

    // Dequeue the top element
    dequeue();

    // Check the top element
    ind = peek();
    cout << pr[ind].value << endl;

      // Dequeue the top element
```

```
    dequeue();

      // Check the top element
    ind = peek();
    cout << pr[ind].value << endl;

    return 0;
}
```

Output

16

14

12

Complexity Analysis:

- *enqueue(): O(1)*
- *peek(): O(N)*
- *dequeue: O(N)*

Application of Priority Queue:

- For Scheduling Algorithms the CPU has to process certain tasks having priorities. The process of having higher priority gets executed first.
- In a time-sharing computer system, the process of waiting for the CPU time gets loaded in the priority queue.
- A Sorting-priority queue is used to sort heaps.