

K-Largest Elements in C++

```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;

void solve(int n, vector<int>& arr, int k) {
    priority_queue<int, vector<int>, greater<int>>
    pq; // Min-heap

    for (int i = 0; i < arr.size(); ++i) {
        if (i < k) {
            pq.push(arr[i]);
        } else {
            if (arr[i] > pq.top()) {
                pq.pop();
                pq.push(arr[i]);
            }
        }
    }

    vector<int> result;
    while (!pq.empty()) {
        result.push_back(pq.top());
        pq.pop();
    }

    for (int j = result.size() - 1; j >= 0; --j) {
        cout << result[j] << " ";
    }
    cout << endl;
}

int main() {
    vector<int> num = {44, -5, -2, 41, 12, 19, 21, -6};
    int k = 2;
    solve(num.size(), num, k);

    return 0;
}
```

Dry Run of `solve(n, arr, k)`

Input:

```
arr = {44, -5, -2, 41, 12, 19, 21, -6};
k = 2;
```

Step 1: Initialize Min-Heap (`priority_queue`)

- Min-heap stores the **top k largest** elements.
- Initial heap (empty):** `pq = {}`

Step 2: Process First `k` Elements (`k = 2`)

Iteration	<code>arr[i]</code>	Heap After Push (<code>pq</code>)
<code>i = 0</code>	44	{44}
<code>i = 1</code>	-5	{-5, 44}

Step 3: Process Remaining Elements

Iteration	<code>arr[i]</code>	Compare With <code>pq.top()</code>	Action Taken	Heap After Update
<code>i = 2</code>	-2	<code>-5 < -2</code>	Pop -5, Push -2	{-2, 44}
<code>i = 3</code>	41	<code>-2 < 41</code>	Pop -2, Push 41	{41, 44}
<code>i = 4</code>	12	<code>41 > 12</code>	No Change	{41, 44}
<code>i = 5</code>	19	<code>41 > 19</code>	No Change	{41, 44}
<code>i = 6</code>	21	<code>41 > 21</code>	No Change	{41, 44}
<code>i = 7</code>	-6	<code>41 > -6</code>	No Change	{41, 44}

Step 4: Extract Elements from Min-Heap

- Extract elements in ascending order:
{41, 44}
- Reverse order to print in descending: **44 41**

Output:

44 41