|  |  |
| --- | --- |
| **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** | **arr[i]** | **Heap After Push (pq)** | | --- | --- | --- | | i = 0 | 44 | {44} | | i = 1 | -5 | {-5, 44} |   **Step 3: Process Remaining Elements**   | **Iteration** | **arr[i]** | **Compare With pq.top()** | **Action Taken** | **Heap After Update** | | --- | --- | --- | --- | --- | | i = 2 | -2 | -5 < -2 | Pop -5, Push -2 | {-2, 44} | | i = 3 | 41 | -2 < 41 | Pop -2, Push 41 | {41, 44} | | i = 4 | 12 | 41 > 12 | No Change | {41, 44} | | i = 5 | 19 | 41 > 19 | No Change | {41, 44} | | i = 6 | 21 | 41 > 21 | No Change | {41, 44} | | i = 7 | -6 | 41 > -6 | 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 | |