|  |  |
| --- | --- |
| **K largest elements in C++** | |
| #include <iostream>  #include <queue>  #include <vector>  using namespace std;  void klargest(vector<int>& arr, int k) {  priority\_queue<int, vector<int>, greater<int>> pq;  // Insert the first k elements into the min heap  for (int i = 0; i < k; i++) {  pq.push(arr[i]);  }  // For each element from k to end of array, check if it's larger than the smallest in the heap  for (int i = k; i < arr.size(); i++) {  if (pq.top() < arr[i]) {  pq.pop();  pq.push(arr[i]);  }  }  // Print the k largest elements  cout << "K largest elements: ";  while (!pq.empty()) {  cout << pq.top() << " ";  pq.pop();  }  cout << endl;  }  int main() {  // Hardcoded input array  vector<int> arr = {5, 15, 10, 20, 8, 25, 18};  int k = 3;  // Call the klargest function to find and print the k largest elements  klargest(arr, k);  return 0;  } | Step-by-Step Dry Run  | **Step** | **i** | **Element** | **Min Heap Before** | **Action** | **Min Heap After** | | --- | --- | --- | --- | --- | --- | | Init | - | - | [] | Start inserting first k=3 |  | | 1 | 0 | 5 | [] | Push 5 | [5] | | 2 | 1 | 15 | [5] | Push 15 | [5, 15] | | 3 | 2 | 10 | [5, 15] | Push 10 | [5, 15, 10] | | 4 | 3 | 20 | [5, 15, 10] | 20 > 5 → pop 5, push 20 | [10, 15, 20] | | 5 | 4 | 8 | [10, 15, 20] | 8 < 10 → do nothing | [10, 15, 20] | | 6 | 5 | 25 | [10, 15, 20] | 25 > 10 → pop 10, push 25 | [15, 20, 25] | | 7 | 6 | 18 | [15, 20, 25] | 18 > 15 → pop 15, push 18 | [18, 25, 20] |  ✅ Final Heap Contents: [18, 25, 20] This heap now contains the **top 3 largest elements**: **18, 25, 20** 🖨️ Output: K largest elements: 18 20 25 |
| K largest elements: 18 20 25 | |