|  |
| --- |
| #include <stdio.h> |
|  | void merge(int arr[], int p, int q, int r) { |
|  | // Create L ? A[p..q] and M ? A[q+1..r] |
|  | int n1 = q - p + 1; |
|  | int n2 = r - q; |
|  | int L[n1], M[n2]; |
|  | for (int i = 0; i < n1; i++) |
|  | L[i] = arr[p + i]; |
|  | for (int j = 0; j < n2; j++) |
|  | M[j] = arr[q + 1 + j]; |
|  | // Maintain current index of sub-arrays and main array |
|  | int i, j, k; |
|  | i = 0; |
|  | j = 0; |
|  | k = p; |
|  | // Until we reach either end of either L or M, pick larger among |
|  | // elements L and M and place them in the correct position at A[p..r] |
|  | while (i < n1 && j < n2) { |
|  | if (L[i] <= M[j]) { |
|  | arr[k] = L[i]; |
|  | i++; |
|  | } else { |
|  | arr[k] = M[j]; |
|  | j++; |
|  | } |
|  | k++; |
|  | } |
|  | // When we run out of elements in either L or M, |
|  | // pick up the remaining elements and put in A[p..r] |
|  | while (i < n1) { |
|  | arr[k] = L[i]; |
|  | i++; |
|  | k++; |
|  | } |
|  | while (j < n2) { |
|  | arr[k] = M[j]; |
|  | j++; |
|  | k++; |
|  | } |
|  | } |
|  | // Divide the array into two subarrays, sort them and merge them |
|  | void mergeSort(int arr[], int l, int r) { |
|  | if (l < r) { |
|  | // m is the point where the array is divided into two subarrays |
|  | int m = l + (r - l) / 2; |
|  | mergeSort(arr, l, m); |
|  | mergeSort(arr, m + 1, r); |
|  | // Mergingthe 2 short arrays |
|  | merge(arr, l, m, r); |
|  | } |
|  | } |
|  | // Print the array |
|  | void printArray(int arr[], int size) { |
|  | for (int i = 0; i < size; i++) |
|  | printf("%d ", arr[i]); |
|  | printf("\n"); |
|  | } |
|  | int main() { |
|  | int arr[] = {6, 5, 12, 10, 8, 1}; |
|  | int size = sizeof(arr) / sizeof(arr[0]); |
|  | mergeSort(arr, 0, size - 1); |
|  | printf("Sorted array: \n"); |
|  | printArray(arr, size); |
|  | } |