Heap Sort Code In C++

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

// Function to heapify a subtree rooted at index i
void heapify(int arr[], int n, int i) {
    int largest = i;         // Initialize largest as root
    int left = 2 * i + 1;    // left child
    int right = 2 * i + 2;   // right child

    // If left child is larger than root
    if (left < n && arr[left] > arr[largest])
        largest = left;

    // If right child is larger than largest so far
    if (right < n && arr[right] > arr[largest])
        largest = right;

    // If largest is not root
    if (largest != i) {
        swap(arr[i], arr[largest]);

        // Recursively heapify the affected subtree
        heapify(arr, n, largest);
    }
}

// Main function to perform heap sort
void heapSort(int arr[], int n) {
    // Build a max heap
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // Extract elements from heap one by one
    for (int i = n - 1; i >= 0; i--) {
        // Move current root to end
        swap(arr[0], arr[i]);
```

```
        // Call max heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

// Function to print an array
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// Driver code
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array: ";
    printArray(arr, n);

    heapSort(arr, n);

    cout << "Sorted array: ";
    printArray(arr, n);

    return 0;
}
```

Output-

Original array: 12 11 13 5 6 7
Sorted array: 5 6 7 11 12 13