```
#include <iostream>

void heapify(int arr[], int size, int root) {
    int largest = root;    // 最大值的索引
    int left = 2 * root + 1;
    int right = 2 * root + 2;

    // 找出左子节点和右子节点中的最大值
    if (left < size && arr[left] > arr[largest]) {
        largest = left;
    }
    if (right < size && arr[right] > arr[largest]) {
        largest = right;
    }

    // 若最大值不是根节点，则交换根节点和最大值，并递归调整堆
    if (largest != root) {
        int temp = arr[root];
        arr[root] = arr[largest];
        arr[largest] = temp;

        heapify(arr, size, largest);
    }
}
```

这是堆调整函数，用于调整以root为根节点的子树成为最大堆。它首先找出根节点、左子节点和右子节点中的最大值，并将最大值与根节点交换。然后，如果最大值发生变化，则递归调用heapify函数继续调整交换后的子树。

```
void heapSort(int arr[], int size) {
    // 构建最大堆
    for (int i = size / 2 - 1; i >= 0; i--) {
        heapify(arr, size, i);
    }

    // 逐个将堆顶元素与最后一个元素交换，并调整堆
    for (int i = size - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        heapify(arr, i, 0);
    }
}
```

这是堆排序函数，它接收一个数组和数组的大小作为参数。首先，它通过从最后一个非叶子节点开始，自底向上构建最大堆。然后，它通过逐个将堆顶元素与最后一个元素交换，并调用heapify函数进行堆调整，保持最大堆的性质。重复这个过程直到整个数组有序。

```
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int size = sizeof(arr) / sizeof(arr[0]);

    heapSort(arr, size);

    std::cout << "排序结果: ";
    for (int i = 0; i < size; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

这是主函数，用于测试堆排序算法。它创建一个包含待排序元素的数组，并计算数组的大小。然后，它调用heapSort函数对数组进行排序。最后，它输出排序结果。

希望这些注释和解释能帮助您理解堆排序算法的思路和实现。