```cpp
#include <iostream>

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;    // 左子数组的大小
    int n2 = right - mid;       // 右子数组的大小

    // 创建临时数组存储左右子数组
    int* leftArr = new int[n1];
    int* rightArr = new int[n2];

    // 将数据复制到临时数组
    for (int i = 0; i < n1; i++) {
        leftArr[i] = arr[left + i];
    }
    for (int j = 0; j < n2; j++) {
        rightArr[j] = arr[mid + 1 + j];
    }

    int i = 0;  // 左子数组的索引
    int j = 0;  // 右子数组的索引
    int k = left;  // 合并后的数组的索引

    // 比较左右子数组的元素，将较小的元素放入合并后的数组
    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }

    // 将左子数组中剩余的元素放入合并后的数组
    while (i < n1) {
        arr[k] = leftArr[i];
        i++;
        k++;
    }

    // 将右子数组中剩余的元素放入合并后的数组
    while (j < n2) {
        arr[k] = rightArr[j];
        j++;
        k++;
    }

    // 释放临时数组的内存
    delete[] leftArr;
    delete[] rightArr;
}
```

这是归并排序中的合并函数。它接收一个数组和三个参数，即左子数组的起始索引、右子数组的起始索引（也是中间索引）和右子数组的结束索引。合并函数的主要任务是将左右两个有序的子数组合并成一个有序的数组。首先，计算左右子数组的大小，并创建临时数组leftArr和rightArr来存储左右子数组的元素。然后，将左右子数组的元素逐个比较，并将较小的元素放入合并后的数组。最后，将左右子数组中剩余的元素放入合并后的数组，并释放临时数组的内存。

```
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;    // 计算中间索引

        // 递归地对左右子数组进行归并排序
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // 合并左右子数组
        merge(arr, left, mid, right);
    }
}
```

这是归并排序函数，它接收一个数组和两个参数，即左子数组的起始索引和右子数组的结束索引。归并排序使用分治的思想，将数组不断划分为更小的子数组，直到子数组的大小为1，然后再将这些子数组逐个合并，得到有序的数组。首先，判断左子数组的起始索引是否小于右子数组的结束索引，如果是，则进行排序操作。首先，计算中间索引。然后，递归地对左右子数组进行归并排序。最后，调用合并函数merge将左右子数组合并为一个有序数组。

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

    mergeSort(arr, 0, size - 1);

    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，用于测试归并排序算法。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mergeSort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归并排序算法的思路和实现。