#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

// 如果一个数列只有一个数，那么它本身就是有序的；如果只有两个数，那么一次比较就可以完成排序

// 归并排序，数量逐层分解到一个数的序列，然后合并，合并需要额外的空间，而快速排序是原地排序

// 分类 -------------- 内部比较排序

// 数据结构 ---------- 数组

// 最差时间复杂度 ---- O(nlogn)

// 最优时间复杂度 ---- O(nlogn)

// 平均时间复杂度 ---- O(nlogn)

// 所需辅助空间 ------ O(n)

// 稳定性 ------------ 稳定

// 合并两个已排好序的数组A[left...mid]和A[mid+1...right]

void Merge(int A[], int left, int mid, int right)

{

int len = right - left + 1;

int \*temp = new int[len]; // 辅助空间O(n)

int index = 0;

int i = left; // 前一数组的起始元素

int j = mid + 1; // 后一数组的起始元素

while(i <= mid && j <= right) // 从小到大依次放到辅助数组中

{

if(A[i] <= A[j])

temp[index++] = A[i++];

else

temp[index++] = A[j++];

}

while(i <= mid) // 对子序列A[left:mid]剩余的依次处理

{

temp[index++] = A[i++];

}

while(j <= right) // 对子序列B[mid+1:right]剩余的依次处理

{

temp[index++] = A[j++];

}

for(int k = 0; k < len; k++) // 将合并后的序列复制到原来的A数组

{

A[left++] = temp[k];

}

}

// 递归实现的归并排序(自顶向下)

void MergeSortRecursion(int A[], int left, int right)

{

if(left == right) // 当待排序的序列长度为1时，递归开始回溯，进行merge操作

return;

int mid =(left + right) / 2;

MergeSortRecursion(A, left, mid); // 对A[left:mid]中的元素排序

MergeSortRecursion(A, mid + 1, right); // 对A[mid:right]中的元素排序

Merge(A, left, mid, right);

}

// 非递归(迭代)实现的归并排序(自底向上)

void MergeSortIteration(int A[], int len)

{

int left, mid, right; // 子数组索引,前一个为A[left...mid]，后一个子数组为A[mid+1...right]

for(int i=1; i<len; i\*=2) // 子数组的大小i初始为1，每轮翻倍

{

left = 0;

while(left + i < len) // 后一个子数组存在(需要归并)

{

mid = left + i - 1;

right = (mid+i)<len?mid+i:len-1; // 后一个子数组大小可能不够

Merge(A, left, mid, right);

left = right + 1; // 前一个子数组索引向后移动

}

}

}

int main()

{

int A1[] = { 6, 5, 3, 1, 8, 7, 2, 4 }; // 从小到大归并排序

int A2[] = { 6, 5, 3, 1, 8, 7, 2, 4 };

int n1 = sizeof(A1) / sizeof(int);

int n2 = sizeof(A2) / sizeof(int);

MergeSortRecursion(A1, 0, n1 - 1); // 递归实现

MergeSortIteration(A2, n2); // 非递归实现

printf("递归实现的归并排序结果：");

for(int i = 0; i < n1; i++)

{

printf("%d ", A1[i]);

}

printf("\n");

printf("非递归实现归并排序结果：");

for(i = 0; i < n2; i++)

{

printf("%d ", A2[i]);

}

printf("\n");

}