# 1 项目简介

## 随机函数产生一定数量的随机数，用不同的的排序方法进行排序，并统计每种排序所花费的排序时间和交换次数（归并排序给出比较次数）。其中，随机数的个数在[2,50000]范围，由用户定义，系统产生随机数。排序算法包括冒泡排序，选择排序，直接插入排序，希尔排序，快速排序，堆排序，归并排序和基数排序八种排序算法。

# 2 程序说明

本程序中通过函数模板将8种排序算法包含在头文件中，生成随机序列之后通过调用头文件中的排序函数实现排序并返回交换、比较次数，通过排序函数执行的起始时间差确定排序用时。

接下来对八种排序算法的逻辑进行说明：

①冒泡排序：

冒泡排序的核心思想是不断比较两个相邻的元素，将值大的元素交换到右边。具体流程是：依次比较相邻的两个数，将比较小的数放在前面，比较大的数放在后面。如此继续，直到比较完最后的两个数，将小数放在前面，大数放在后面，那么一次遍历完之后，最后一个数一定是数组中最大的一个数，正如冒泡一般，所以在比较第二趟的时候，最后一个数是不参加比较的。此后，依次类推，每一趟比较次数减少依次，最终便可实现所有数据从小到大的排列。

需要注意的一点是：如果在排序中途出现某次遍历没有交换的情况，那么说明此时整个序列已经构成从小到大的有序序列，已经可以结束排序，而不用一直等到遍历总次数等于数组长度。

②选择排序：

选择排序的核心思想是：如果从下到大排列的话，结果序列的第i个位置应该存放当前序列第i个最小值，所以需要通过遍历与比较从当前序列中找到第i小的对象放在结果序列的第i个位置。

具体流程是：主体用循环结构实现，从i等于0开始，先假定第i个最小值的位置min等于i，接着从i+1开始遍历数组，如果遇到比min位置更小的数，就更新min值，遍历完之后，如果min不等于最初的i，进行交换并计数。以此类推，当循环到i等于数组长度减一，则数组排序完成。

③直接插入排序：

直接插入排序的核心思想是：当插入第i(i>=1)个对象时，前面的数据已经排好序。这时，将第i个对象的关键码和前面已经排好序的数据进行比较，找到合适的插入位置将其插入即可，而原来位置上的对象向后顺延。

具体流程是：从i=1开始，首先选取第1个元素作为待插入元素暂存，接着令j=i，从j开始从后往前遍历，定位插入位置并进行交换、计数，最终将暂存的元素插入相应位置。接着i自增，一直循环执行上述操作，直到i等于数字长度减一，结束循环。

④希尔排序：

希尔排序其实是对直接插入排序的优化，我们可以考虑这样一种极端情况：当长度为1000的数组，前面有序区域的数组长度为999，此时我们用第1000个数去跟前面有序区域的所有元素比较大小，但恰巧第1000个数又是这1000个数里最小的，它本应该在索引为1的位置。那么前面有序区域里的999个元素都要往后移动一个位置，这种情况就非常影响排序性能。

所以，为了规避这种极端情况，希尔排序的核心思想是：设待排序对象序列有n 个对象，首先取一个整数 gap < n 作为间隔，将全部对象分为 gap 个子序列，所有距离为 gap 的对象放在同一个子序列中，在每一个子序列中分别施直接插入排序。然后缩小间隔 gap，重复上述的子序列划分和排序工作。直到最后取 gap == 1 ，将所有对象放在同一个序列中排序为止。

希尔排序具体的代码逻辑如下：主体采用循环结构实现，gap不为0则一直循环，进行排序操作，之后在通过gap划分的不同数字段中执行直接插入排序：先暂存待插入元素，之后通过循环比较，定位插入位置并交换并计数，最后将暂存的元素插入相应位置。

⑤快速排序：

快速排序的核心思想是：通过一趟排序将待排记录分隔成独立的两部分，其中一部分记录的关键字均比另一部分的关键字小，则可分别对这两部分记录继续进行排序，以达到整个序列有序。

快速排序的具体流程是：(递归思路)

首先设置两个变量 left、right，排序开始时：left=0，right=num-1。接着确定整个数组的基准位置，所有元素比基准值小的摆放在基准前面，所有元素比基准值大的摆在基准的后面。本程序中默认数组的第一个数为基准数据，赋值给pivot，即pivot=array[left]。因为默认数组的第一个数为基准，所以接下来从left+1开始向后搜索，直到到达右边界，把比基准数小的数字调到左边位置，相应的，较大的数字调到右边位置，而且最后把基准数移动到这两个序列的中间位置，并计数交换次数。接下来递归调用快速排序函数，排序分界点前和分界点后的子数组排序，最终就会得到排序好的数组。

⑥堆排序：

堆排序的核心思想是：利用堆的向下调整算法将待排序序列形成最大堆，接着通过一系列的对象交换并重新调整堆，进行排序。

堆排序的具体流程是：

1. 首先调用FilterDown函数，设子节点位置为child，则只要child没有大于数字总长度，则一直执行下面的操作：比较两个子节点大小，选择最大的子节点，并更新child。接着判断父节点是否小于最大子节点，若小于，则父节点与最大子节点交换位置，并计数。这样一来，就可以形成局部的最大堆：父节点比两个子节点值大。所以，配合循环结构，将整个数组调整成最大堆即可生成初始最大堆。
2. 将堆顶元素与末尾元素进行交换，使末尾元素最大。然后继续调整堆，再将堆顶元素与末尾元素交换，得到第二大元素。如此反复进行交换、重建、交换，最终即可得到从小到大的有序排序序列。

⑦二路归并排序：

二路归并排序的核心思想是：对序列的元素进行逐层折半分组，然后从最小分组开始比较排序，合并成一个大的分组，逐层进行，最终所有的元素都是有序的，具体来看：假设初始对象序列有 n 个对象，首先把它看成是 n 个长度为 1 的有序子序列 (归并项 )，先做两两归并，得到⎡n / 2 ⎤ 个长度为 2 的归并项 ( 如 果 n 为奇数，则最后一个有序子序列的长度为1)；再做两两归并；再做两两归并，如此重复，最后得到 ，一个长度为 n 的有序序列。

具体代码流程是：

1. 通过Merge函数归并len长度有序序列中数字大小顺序

通过循环结构，不断比较两个待归并序列，选择较小的数写入结果序列，此时，该较小数原序列中游标记号后移一位，直到某一个序列达到边界。之后判断是否存在某个序列中的遍历位置未达到边界的情况，若存在，则说明剩余的序列都比当前结果序列中的最后一个数字大，且未排完的部分是有序序列，直接将剩余的数字序列全部写入结果序列中即可。

需要注意的是，一开始的序列长度是1，之后序列长度不断扩大两倍，这样的序列长度划分方式保证了，每次待归并序列都是有序的。

1. 通过MergePass函数完成一趟归并排序

设数组initArray和mergedArray中的n个对象已经分成长度为len的归并项，通过循环结构将这些待归并项两两归并，形成一些长度为len\*2的待归并项，结果存放在结果数组中。

需要注意的一点是，如果n不是2\*len的整数倍，则一趟归并到最后，可能遇到两种情形：

剩下一个长度为len的归并项和另一个长度不足len的归并项，这是可再利用merge函数，将它们归并成一个长度小于2\*len的归并项；

只剩下一个长度不超过len的待归并项，由于归并的比较原则以及此时待归并项内部本身的序列是有序的，只要直接将剩余的未归并部分紧接着写入结果序列即可。

1. 调用MergeSort函数进行完整的归并排序

首先初始化最开始的len为1,接着进入循环结构，只要len<num，就一直循环。在每一次循环中调用两次MergePass进行归并排序，且每次之后都使len长度扩大为两倍。只是这两次中，第一次是将temp\_1作为初始序列，temp\_2作为结果序列，第二次归并中，二者的位置互换（其中，temp\_1与temp\_2一开始初始状态下都是原始待排序序列的复制，之所以不直接改动原待排序序列是因为本程序中所有的排序算法都公用一个原始序列，所以每个函数直接改动的都是原始序列的副本），这样一来，最终的排序序列就暂存在temp\_1中。

⑧基数排序：

基数排序的核心思想是：通过分配与收集的方式对多关键码进行排序进而实现对数据序列的排序。在本程序中，个位、十位、百位等每一位都存在0-9共10个关键码，首先对按照个位排序，接着再按照十位排序，以此类推，直到对最高位排完序，那么整个数字序列就自然地形成了从小到大的有序序列。

本程序中，基数排序的代码流程是：

整体上采用循环结构来实现，这里不妨使用for循环，在for循环中我们需要做几件事情，因为我们是利用桶的思想来进行排序，要将元素放入到对应的不同的桶中，对于每一次排序我们都需要清空桶，也就是将0-9这10个桶全部置为0，个位排完置为0，重新计数，十位排完置为0，也重新计数，以此类推。置为0之后，我们需要统计一下，当前这一轮排序，每一个桶中相应放进了多少个元素。之后，通过0-9共10个关键码中各关键码对应的数字的数量进行统计，并依据此将tmp（注：tmp是用来暂存排序中间过程的，长度与原始待排序序列长度一样的数组）中的位置依次分配给每个桶，并将所有桶中记录依次收集到tmp中，最后将tmp中的内容复制到array\_copy（注：array\_copy可以视为结果数组）中。每次循环都进行这样的过程，直到将最高位排完序，循环结束。此时，原始序列已经形成了从小到大的排序序列。

# 3. 程序执行流程

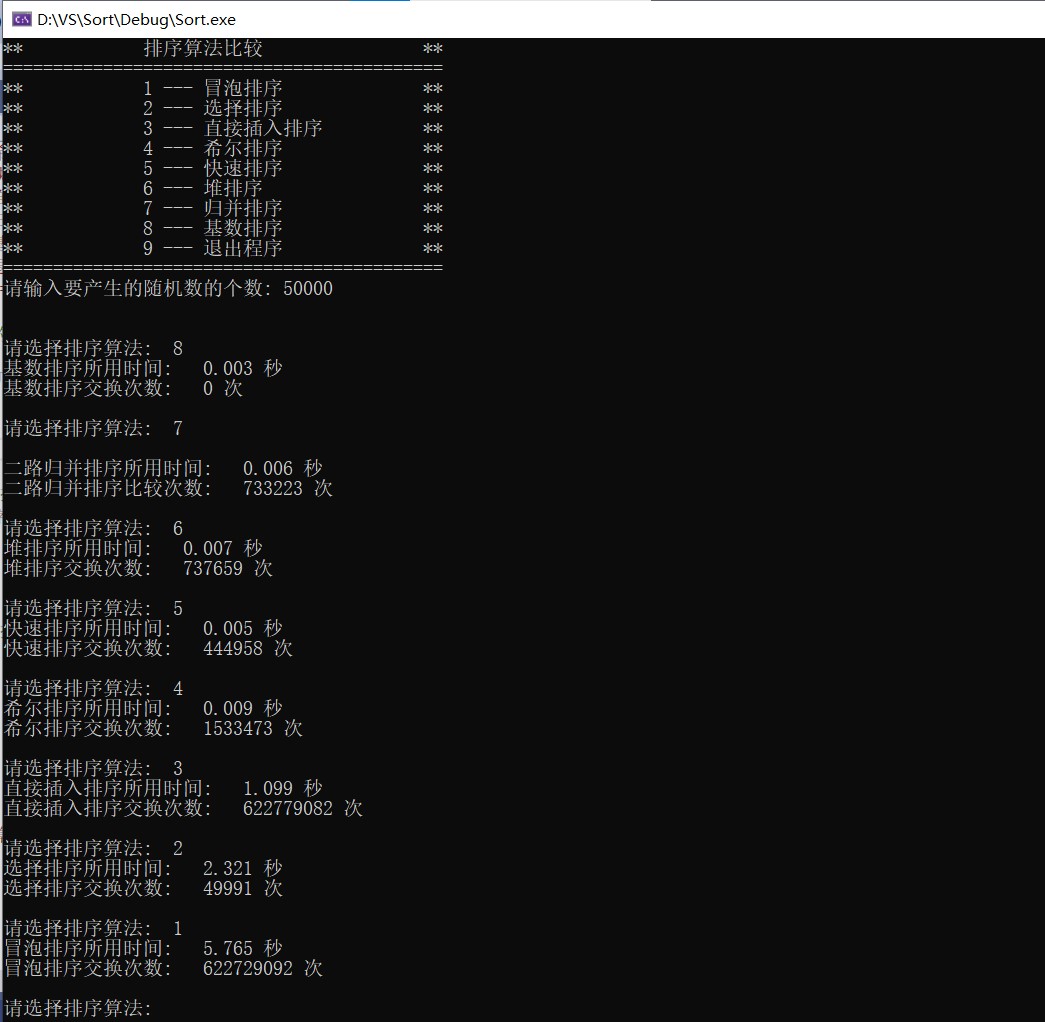
本程序先有用户输入排序序列的数字规模num，之后通过random函数配合循环结构，生成num个[0,num]的随机数，之后根据用户选择的排序算法，调用不同的排序函数进行排序，并输出交换次数、排序时间等信息。

需要注意的是，调用random函数之前务必要通过

srand(static\_cast<unsigned int>(time(NULL)));使用当前时间进行随机数发生器的初始化，使每次生成的随机序列不一样。

# 4.测试

# ①50000个随机数，分别调用不同的排序算法进行排序：



②对非法输入的处理：生成随机数个数输入非法或超范围，选择的排序算法无效

