哈尔滨工程大学

实 验 报 告

实 验 名 称：­­

班 级：

学 号：

姓 名：

实 验 时 间：

成 绩：

指 导 教 师：

实验室名称：

**哈尔滨工程大学实验室与资产管理处 制**

|  |
| --- |
| 实验名称：直接插入排序，快速排序，直接选择排序验证 |
| 一、问题描述  进行以下三种排序方式的验证性实验  ·直接插入排序  ·快速排序  ·直接选择排序  二、数据结构设计  采用数组的形式进行数据元素的存储  const int N = 20;  int arr[N]; //原数组  int brr[N]; //复制原数组，避免对原数组的直接操作  三、算法设计    1.随机生成数组，避免手动输入的麻烦  void rand\_arr(){  for(int i = 0; i < N; ++i){  int tmp;  while(1){  int flag = 0;  tmp = rand() % N;  for(int j = 0; j < i; ++j){  if(tmp == arr[j]){  flag = 1;  break;  }  }  if(flag == 0){  arr[i] = tmp;  break;  }  }  }  }  2.复制数组元素  void init\_array(){  for(int i = 0; i < N; ++i){  brr[i] = arr[i];  }  }  3.顺序输出当前状态的数组元素排列方式  void display(int \*a, int size){  for(int i = 0; i < size; ++i){  printf("%5d", a[i]);  }  cout << endl;  }  4.直接插入排序  插入排序通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入，如此重复，直至完成序列排序。  <1>从序列第一个元素开始，该元素可以认为已经被排序  <2>取出下一个元素，设为待插入元素，在已经排序的元素序列中从后向前扫描，如果该元素(已排序)大于待插入元素，将该元素移到下一位置。  <3>重复步骤2，直到找到已排序的元素小于或者等于待排序元素的位置，插入元素  <4>重复2，3步骤，完成排序。  void InsertSort(int \*array, int size){  for(int i = 1; i < size; ++i){  int tmp = array[i];  int j = i - 1;  while(j >= 0 && array[j] > tmp){  array[j + 1] = array[j];  --j;  }  array[j + 1] = tmp;  cout << “第” << i << “步:\t”;  display(array, size);    }  }  5.快速排序  <1>从数列中挑出一个元素，称为基准  <2>重新排序数列，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(相同的数可以到任一边)。在这个分区退出之后，该基准就处于数列的中间位置  <3>递归地把小于基准值元素的子数列和大于基准值元素的子数列排序  void QuickSort(int\* array, int left, int right){  if(left >= right)  return;  int i, j, base, tmp;  i = left;  j = right;  base = array[left];  while (i < j){  while (array[j] >= base && i < j)  j--;  while (array[i] <= base && i < j)  i++;  if(i < j){  tmp = array[i];  array[i] = array[j];  array[j] = tmp;  }  }  array[left] = array[i];  array[i] = base;  cout << “第” << ++t << “步:\t”;  display(brr, N);    QuickSort(array, left, i - 1);  QuickSort(array, i + 1, right);  }  6.直接选择排序  <1>找到数组中最小的元素，将它和数组的第一个元素交换位置。  <2>第二步，在剩下的元素中继续寻找最小的元素，和数组的第二个元素交换位置。  <3>如此循环，直到整个数组排序完成  void SelectSort(int\* array, int n){  t = 0;  for(int i = 0; i < n; ++i){  int minn = i;  for(int j = i + 1; j < n; ++j){  if(array[j] < array[minn]){  minn = j;  }  }  if(minn != i){  int tmp = array[i];  array[i] = array[minn];  array[minn] = tmp;  }  cout << “第” << ++t << “步:\t”;  display(array, N);    }  cout << endl;  }  四、界面设计  ·随机生成元素序列并显示  ·分别按步骤显示三种排序方式的每步操作    五、运行测试与分析  样本数为N = 15：    六、实验收获与思考  本次实验我选择了对三种排序方法的验证性实验，虽然是最简单的实验，但通过对三种排序方法的验证，我收获了许多知识和对问题的思考，这远大于做综合性实验带给我的收获。通过本次实验，我更加深入的了解了三种排序方式的原理和具体实现的方法，给我很深的启发和思考。 |