**《数据结构综合设计》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | 人工智能与大数据学院 | 专业 | 虚拟现实技术 | 班级 | 21级3班 | 学生姓名 | 解芸芸 |
| 实验  周次 | 13-14 | 实验  日期 | 2023.5.25 | 学时 | 6 | 教师姓名 | 李昊康 |
| 项目名称 | | 排序的应用 | | | | | |
| 实验  类别 | 🗹验证型实验 🞎设计型实验 🞎综合型实验 🞎其它 | | | | | 成绩：93 | |
| 1. 实验目的及具体要求   实验目的：  1.实现多种类型的排序算法（插入排序、交换排序、选择排序、归并排序等）；  2.理解排序过程；  3.计算比较次数和移动次数，对比分析算法性能的优劣与适用场景；  具体要求：  编写程序实现插入排序、希尔排序、冒泡排序、快速排序、简单选择排序、堆排序。中任意3种。   1. 实验仪器、设备和材料   硬设备：PC机  软件环境：Windows VS2019   1. 实验内容、步骤及实验数据记录 2. 插入排序   #include<iostream>  using namespace std;  void print(int a[], int n)  {  for (int j = 0; j < n; j++)  {  cout << a[j] << " ";  }  cout << endl;  }  void insertSort(int a[], int n)  {  for (int i = 1; i < n; i++) //第一个元素作为基准元素，从第二个元素开始把其插到正确的位置  {  if (a[i] < a[i - 1]) //如果第i个元素比前面的元素小  {  int j = i - 1; //需要判断第i个元素与前面的多个元素的大小，换成j继续判断  int x = a[i]; //将第i个元素复制为哨兵  while (j >= 0 && x < a[j]) //找哨兵的正确位置，比哨兵大的元素依次后移  {  a[j + 1] = a[j];  j--;  }  a[j + 1] = x; //把哨兵插入到正确的位置  }  }  }  int main()  {  int a[10] = {0,4,10,9,7,8,5,6,8,2};  cout << "初始序列：";  print(a, 10);  insertSort(a, 10);  cout << "排序结果：";  print(a, 10);  system("pause");  }  运行结果：     1. 快速排序   #include<iostream>  using namespace std;  void quickSort(int a[], int, int);  int main()  {  int array[] = { 34,65,12,43,67,5,78,10,3,70 }, k;  int len = sizeof(array) / sizeof(int);  cout << "初始序列:" << endl;  for (k = 0; k < len; k++)  cout << array[k] << ",";  cout << endl;  quickSort(array, 0, len - 1);  cout << "排序结果:" << endl;  for (k = 0; k < len; k++)  cout << array[k] << ",";  cout << endl;  system("pause");  return 0;  }  void quickSort(int s[], int l, int r)  {  if (l < r)  {  int i = l, j = r, x = s[l];  while (i < j)  {  while (i < j && s[j] >= x) // 从右向左找第一个小于x的数  j--;  if (i < j)  s[i++] = s[j];  while (i < j && s[i] < x) // 从左向右找第一个大于等于x的数  i++;  if (i < j)  s[j--] = s[i];  }  s[i] = x;  quickSort(s, l, i - 1); // 递归调用  quickSort(s, i + 1, r);  }  }  运行结果：     1. 冒泡排序   #include <iostream>  using namespace std;  void maopao(int a[], int n) {  int temp, i, j;  for (i = 0; i < n - 1; i++) {  for (j = 0; j < n - i - 1; j++) {  if (a[j] > a[j + 1]) {  temp = a[j];  a[j] = a[j + 1];  a[j + 1] = temp;  }  }  }  }  int main() {  int i, n;  int a[100];  cout << "请输入数组总数：" << endl;  cin >> n;  cout << "请输入一组整数：" << endl;  for (i = 0; i < n; i++) {  cin >> a[i];  }  maopao(a, n);  cout << "排序后的数组为：" << endl;  for (i = 0; i < n; i++) {  cout << a[i] << " ";  }  cout << endl;  }  运行结果：    分析：  1.插入排序的基本操作就是将一个数据插入到已经排好序的有序数据中，从而得到一个新的、个数加一的有序数据，算法适用于少量数据的排序，是稳定的排序方法。2.快速排序是对冒泡排序的一种改进。通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以[递归](https://baike.so.com/doc/5725016-5937752.html" \t "https://baike.so.com/doc/_blank)进行。3.冒泡排序是[空间复杂度](https://so.csdn.net/so/search?q=%E7%A9%BA%E9%97%B4%E5%A4%8D%E6%9D%82%E5%BA%A6&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/SUYDFBGUSDYFBLS/article/details/_blank)低，时间复杂度高。升序排序中每一轮比较会把最大的数下沉到最低，所以相互比较的次数每一轮会比前一轮少一次，是一种稳定的[排序算法](https://so.csdn.net/so/search?q=%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/SUYDFBGUSDYFBLS/article/details/_blank)。 | | | | | | | |

说明：1. 实验周次：填写实际上课周，如第5-8周上课填“5-8”或第10周上课填“10”。

1. 实验报告各部分内容需详实填写，按实验指导书上的评分标准给出分数。
2. 实验类型参考实验类型说明文件。