**《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2班 | 学生姓名 | 宋熙桂 |
| 实验  周次 | 13-14 | 实验  日期 | 2023.5.26 | 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：91 | |
| 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   实验代码：  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;  }  结果：    2.快速排序：  #pragma warning(disable : 4996)  #pragma warning(disable : 6031)  #include <iostream>  #include<stdio.h>  #include<string.h>  #include<stdlib.h>  using namespace std;  const int N = 1e6 + 10;  int q[N];  int n;  void quick\_sort(int q[], int l, int r) {    if (l >= r) return;  int i = l - 1, j = r + 1, x = q[l + r >> 1];  while (i < j) {  do i++; while (q[i] < x);  do j--; while (q[j] > x);  if (i < j) swap(q[i], q[j]);  }  quick\_sort(q, l, j);  quick\_sort(q, j + 1, r);  }  int main() {  printf("请输入个数");  scanf("%d", &n);  printf("请输入所排列数");  printf(":");  for (int i = 0; i < n; i++) scanf("%d", &q[i]);  quick\_sort(q, 0, n - 1);    for (int i = 0; i < n; i++) printf("%d ", q[i]);  return 0;  }  结果：    3.简单选择排序：  #include <iostream>  #include <stdlib.h>  using namespace std;  void selectionSort(int arr[], int len) {  int cFound = 0;  for (int i = len - 1; i >= 0; --i) {  int maxIndex = i;  int maxVal = -1;  for (int j = i; j >= 0; --j) {  if (maxVal < arr[j]) {  maxIndex = j;  maxVal = arr[j];  }  }  int tmp = arr[i];  arr[i] = arr[maxIndex];  arr[maxIndex] = tmp;  }  }  int main() {  int arr[] = { 6,4,2,8,1,3,9,5,7 };  int len = 9;  selectionSort(arr, len);  for (int i = 0; i < len; ++i) {  cout << arr[i] << endl;  }  system("pause");  return 0;  }  结果：    分析：   1. 冒泡排序：是一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果它们的顺序错误就把它们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。 2. 快速排序：在平均状况下，排序 n 个项目要 Ο(nlogn)次比较。在最坏状况下则需要 Ο(n2) 次比较，但这种状况并不常见。事实上，快速排序通常明显比其他 Ο(nlogn) 算法更快，因为它的内部循环（inner loop）可以在大部分的架构上很有效率地被实现出来。 3. 简单选择排序：表现最稳定的排序算法之一，因为无论什么数据用到它的时候，数据规模越小越好。唯一的好处可能就是不占用额外的内存空间了吧。理论上讲，选择排序可能也是平时排序一般人想到的最多的排序方法了吧。 | | | | | | | |

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。