山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号： | 姓名： | | 班级： |
| 实验题目：  1、创建排序类。  2、提供操作选择排序、冒泡排序、插入排序、按名次（计数）排序、快速排序、归并排序。  3、能够显示各种排序算法的中间过程。 | | | |
| 实验学时：60分钟 | | 实验日期： 2018.10.5 | |
| 实验目的：掌握各种排序方法的实现思想。 | | | |
| 硬件环境： 笔记本电脑 | | | |
| 软件环境：win10+Visual Studio 2017 | | | |
| 实验内容：  基本代码如下：  #include<iostream>  using namespace std;  //选择排序  void selectSort(int a[], int n)  {  int i = 0, j = 0, k = 0;  for (i = 0; i < n - 1; i++)  {  k = i;  for (j = i + 1; j < n; j++)  {  if (a[k] > a[j])  {  k = j;  }  }//选择一个值作为标杆，然后循环找到除这个值外最小的值（查找小于标杆的最小值）  int temp;  if (k != i)  {  temp = a[k];  a[k] = a[i];  a[i] = temp;  }//交换这两个值  cout << "排序结果: ";  for (int t = 0; t < 10; t++)  cout << a[t] << " ";  cout << endl;//输出每次排序结果  }  }  //插入排序  void insertSort(int a[], int n)  {  int i, j;  for (i = 1; i < n; i++)  {  for (j = i - 1; j >= 0; j--)  {  if (a[i] > a[j])  break;//比较a[i]与a[j]的大小，确定a[i]的插入位置  }  int temp = a[i];  for (int k = i - 1; k > j; k--)  a[k + 1] = a[k];  a[j + 1] = temp;  //插入a[i]，并将之后的元素后移  cout << "排序结果: ";  for (int t = 0; t < 10; t++)  cout << a[t] << " ";  cout << endl;//输出每次排序结果  }  }  //冒泡排序  void bubbleSort(int a[],int n)  {  int i, j, k, temp;  for (i = 0; i < n; i++)  {  for (j = 0; j < n - i-1; j++) //决定几轮冒泡  {  if (a[j] > a[j + 1])  {  temp = a[j];  a[j] = a[j + 1];  a[j + 1] = temp;  }//如果前面的大于后面的，交换顺序  cout << "排序结果: ";  for (k = 0; k < 10; k++)  cout << a[k] << " ";  cout << endl;//输出每次排序结果  };  };  }  //快速排序  void quickSort(int a[], int leftEnd, int rightEnd)  {  if (leftEnd >= rightEnd)//先判断一下当前待排序数组元素个数是不是大于1,否则就没有必要排序  {  return;//直接退出,没有必要进行排序  }  int piovt = a[leftEnd];//首先我们先保存left索引对应的数据,当前数据作为切割数组的轴  int leftCursor = leftEnd;  int rightCursor = rightEnd;//定义临时变量保存数组两端的索引  while (leftCursor < rightCursor)  {  while (leftCursor < rightCursor)//现在我们通过循环从右边开始搜索一个比轴值小的数据  {  if (piovt <= a[rightCursor])//如果右边的数大于当前的参数轴值  {  rightCursor--;//右端索引指示器左移  }  else//说明我们右端出现比轴值更大的数据  {  a[leftCursor] = a[rightCursor];  leftCursor++;//这个时候我们就可以把这个更大的数据填充到索引轴索引对应的地方  break;  }  }  while (leftCursor < rightCursor)//从左边开始搜索一个比轴值更大的数填写上次留下的空缺  {  if (piovt >= a[leftCursor])  {  leftCursor++;  } // 如果左边的数据小于轴值，我们索引指示器就往右走  else  {  a[rightCursor] = a[leftCursor];  rightCursor--;//说明我们在左端找到了比轴值更大的数据  break;  }  }  cout << "排序结果: ";  for (int t = 0; t < 10; t++)  cout << a[t] << " ";  cout << endl;//输出每次排序结果  }  a[leftCursor] = piovt;  quickSort(a, leftEnd, leftCursor - 1);  quickSort(a, rightCursor + 1, rightEnd);  }  //按名次（计数）排序  void countSort(int a[], int n)  {  int i, j, k;  int rank = 0;  int min, max;  min = max = a[0];  //查找最大值和最小值，可改为三目运算  for (i = 1; i < n; i++)  {  if (a[i] < min)  min = a[i];  else  min = min;  if (a[i] > max)  max = a[i];  else  max = max;  }  //计算k值  k = max - min + 1;  int \*B = new int[k];//创建一个动态空间  for (i = 0; i < k; i++)  B[i] = 0;  for (i = 0; i < n; i++)  B[a[i] - min]++;//这里需要一个偏移量min，因为数组是从下标0开始的，统计次数  for (i = min; i <= max; i++)  for (j = 0; j < B[i - min]; j++)  {  a[rank++] = i;//按元素名次进行排序  cout << "排序结果：";  for (int i = 0; i < n; i++)  cout << a[i] << " ";  cout << endl;  }  delete[] B;//注意要删掉动态空间  }  //归并排序  void merge(int c[], int d[], int startOfFirst, int endOfFirst, int endOfSecond)  {  //把两个相邻的数据段从c归并到d  int first = startOfFirst, //第一个数据段的索引  second = endOfFirst + 1, //第二个数据段的索引  result = startOfFirst; //第三个数据段的索引  //直到有一个数据段归并到归并段d  while ((first <= endOfFirst) && (second <= endOfSecond))  if (c[first] <= c[second])  d[result++] = c[first++];  else  d[result++] = c[second++];  //归并剩余元素  if (first > endOfFirst)  for (int q = second; q <= endOfSecond; q++)  d[result++] = c[q];  else  for (int q = first; q <= endOfFirst; q++)  d[result++] = c[q];  }  void mergePass(int x[], int y[], int n, int segmentSize)  {  //从x到y归并相邻的数据段  int i = 0;//下一个数据段的起点  while (i <= n - 2 \* segmentSize)  {  //从x到y归并相邻的数据段  merge(x, y, i, i + segmentSize - 1, i + 2 \* segmentSize - 1);  i = i + 2 \* segmentSize;  }  //少于两个满数据段  if (i + segmentSize < n)  //剩有两个数据段  merge(x, y, i, i + segmentSize - 1, n - 1);  else  //只剩一个数据段，复制到y  for (int j = i; j < n; j++)  y[j] = x[j];  }  void mergeSort(int a[], int n)  {  //使用归并排序方法对a[0:n-1]排序  int \*b = new int[n];  int segmentSize = 1;  while (segmentSize < n)  {  mergePass(a, b, n, segmentSize);//从a到b的归并  segmentSize += segmentSize;  mergePass(b, a, n, segmentSize);//从b到a的归并  segmentSize += segmentSize;  cout << "排序结果: ";  for (int t = 0; t < 10; t++)  cout << a[t] << " ";  cout << endl;//输出每次排序结果  }  delete[] b;  }  int main()  {  int n;  for (int i = 0; i < 6; i++)  {  n =i; //此处循环目的在于让switch的每个case都可以执行一次  switch (n)  {  case 0:  {  cout << "选择排序：" << endl;  int a[10];  cout << "请输入10个数字: " << endl;  for (int i = 0; i < 10; i++)  cin >> a[i];//输入数组值  selectSort(a, 10);  cout << "最终结果：";  for (int i = 0; i < 10; i++)  cout << a[i] << " ";  cout << endl;  break;  }  case 1:  {  cout << "插入排序：" << endl;  int a[10];  cout << "请输入10个数字: " << endl;  for (int i = 0; i < 10; i++)  cin >> a[i];//输入数组值  insertSort(a, 10);  cout << "最终结果：";  for (int i = 0; i < 10; i++)  cout << a[i] << " ";//输出最终结果  cout << endl;  break;  }  case 2:  {  cout << "快速排序：" << endl;  int a[10];  cout << "请输入10个数字: " << endl;  for (int i = 0; i < 10; i++)  cin >> a[i];//输入数组值  quickSort(a, 0, 9);  cout << "最终结果：";  for (int i = 0; i < 10; i++)  cout << a[i] << " ";  cout << endl;//输出最终结果  break;  }  case 3:  {  cout << "名次排序：" << endl;  int a[10];  cout << "请输入10个数字: " << endl;  for (int i = 0; i < 10; i++)  cin >> a[i];//输入数组值  countSort(a, 10);  cout << "最终结果：";  for (int i = 0; i < 10; ++i)  cout << a[i] << " ";  cout << endl;  break;  }  case 4:  {  cout << "归并排序：" << endl;  int a[10];  cout << "请输入10个数字: " << endl;  for (int i = 0; i < 10; i++)  cin >> a[i];//输入数组值  mergeSort(a, 10);  cout << "最终结果：";  for (int i = 0; i < 10; i++)  cout << a[i] << " ";//输出最终结果  cout << endl;  break;  }  case 5:  {  cout << "冒泡排序：" << endl;  int a[10];  cout << "请输入10个数字: " << endl;  for (int i = 0; i < 10; i++)  cin >> a[i];//输入数组值  bubbleSort(a, 10);  cout << "最终结果：";  for (int i = 0; i < 10; i++)  cout << a[i] << " ";//输出最终结果  cout << endl;  break;  }  default:  {  break;  }  }  }  return 0;  } | | | |
| 运行结果与体会：  体会：  此程序看似是一个，其实是由六个程序组合而来，一开始我将数组输入代码放在循环外，发现只有第一个case是对输入的数组进行排序，其余的case是对第一个case的结果进行排序的，到现在还是不太明白原因何在。于是我采取了将数组输入代码放进循环内的做法，自此问题迎刃而解。现在想来，可能是第一个case的执行改变了数组的值，但究竟是不是此原因，尚待研究。  运行结果：  图片包含 室内  已生成高可信度的说明  图片包含 电子产品  已生成高可信度的说明  图片包含 人员  已生成高可信度的说明 | | | |