中北大学软件学院

**实验报告**

|  |  |
| --- | --- |
| 专 业： | 软件工程 |
| 课程名称： | 数据结构与算法 |
| 班 级： |  |
| 学 号： |  |
| 姓 名： |  |
| 辅导教师： | 李华玲 |

2019年03月制

成绩：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验时间 | 2019年05月 日 时至 时 | 学时数 | 2学时 | |
| 1.实验名称  排序 | | | | |
| 2.实验目的  （1）掌握插入排序算法（重点掌握直接插入排序）及其特点；  （2）掌握快速排序算法及其特点；  （3）掌握归并排序算法及其特点； | | | | |
| 3.实验内容  **基本内容：**  采用顺序存储结构，完成顺序表的创建，实现对顺序表的直接插入排序、冒泡排序。  **选作内容：**  在此基本内容的基础上，实现对顺序表的快速排序或别的排序。 | | | | |
| 4.实验原理  （1）直接插入排序的基本思想是：通过一趟排序将待排序的当前记录插入到前面已经排好序的有序序列中，从而得到一个新的、记录数增1的有序序列。整个排序过程可简述为：先将序列中的第一个记录看成是一个有序的子序列，然后从第二个记录起逐个插入，直至整个序列有序。  （2）快速排序是对起泡排序的一种改进，其基本思想是：通过一趟排序将带排记录分割成独立的两部分，其中一部分记录的关键字都比另一部分记录的关键字小，然后再分别对这两部分继续进行排序，以达到整个序列有序；  （3）（2-路）归并排序是基于归并操作（即将两个有序序列组合成一个新的有序序列）实现排序。这里将含有n个记录的初始序列看成是n个有序的子序列，每个子序列长度为1，然后开始两两归并，直至得到一个长度为n的有序序列为止； | | | | |
| 5.实验源代码  #include<stdio.h>  #include<stdlib.h>  #define MAXSIZE 50  #define EQ(a,b) ((a)==(b))  #define LT(a,b) ((a)< (b))  #define LQ(a,b) ((a)<=(b))  typedef int KeyType;  typedef struct{  KeyType key; //关键字项  }RedType;  typedef struct{  RedType r[MAXSIZE + 1]; //r[0]哨兵  int length; //顺序表长度  }SqList;  void CreateList(SqList &L){  int i;  int length;  printf("请输入线性表的长度：\n");  scanf("%d",&length);  printf("请输入长度为%d的顺序表的内容：\n",length);  for(i=1;i<=length;i++)  {  scanf("%d",&L.r[i].key);  }  L.length=length;  }  //输出顺序表  void outSqList(SqList &L){  int i;  printf("该顺序表如下：\n");  for(i=1;i<=L.length;i++)  printf("%5d",L.r[i].key);  printf("\n");  }  //直接插入排序  void InsertSort (SqList &L){  int i;  int j;  for(i=2;i<=L.length;i++)  if(LT(L.r[i].key,L.r[i-1].key))  {  L.r[0]=L.r[i];  L.r[i]=L.r[i-1];  for(j=i-2;LT(L.r[0].key,L.r[j].key);j--)  L.r[j+1]=L.r[j];  L.r[j+1]=L.r[0];  }  }  //冒泡排序  void BubbleSort(SqList &L)  {  int i, j, change;  RedType temp;  for(i = L.length, change = 1; i > 1 && change; --i)  {  change = 0;  for (j = 1; j < i; ++j)  if (LT(L.r[j+1].key, L.r[j].key))  {  temp = L.r[j];  L.r[j] = L.r[j+1];  L.r[j+1] = temp;  change = 1;  }  }  }  //快速排序  int Partition(SqList &L,int low,int high){  int pivotkey;  L.r[0]=L.r[low];  pivotkey = L.r[low].key;  while(low<high){  while(low<high&&L.r[high].key>=pivotkey) --high;  L.r[low]=L.r[high];  while(low<high&&L.r[low].key<=pivotkey)++low;  L.r[high]=L.r[low];  }  L.r[low]=L.r[0];  return low;  }  void quicksort(SqList L,int low,int high){  int pivotpos,i;  if(low<high)  {  pivotpos=Partition(L,low,high);  quicksort(L,low,pivotpos-1);  quicksort(L,pivotpos+1,high);  }  }  int main()  {  SqList L;  int i,flag;  while(1)  {  printf("\n");  printf("主菜单如下： \n");  printf(" 1.顺序表的创建： \n");  printf(" 2.顺序表的显示： \n");  printf(" 3.对顺序表的直接插入排序： \n");  printf(" 4.对顺序表的冒泡排序 ： \n");  printf(" 5.顺序表的快速排序： \n");  printf(" 6.退出系统 \n");  printf("请输入您要完成的功能序号：\n");  scanf("%d",&flag);  switch(flag)  {  case 1: CreateList(L);  printf("创建成功\n");  break;  case 2: outSqList(L);  break;  case 3: InsertSort(L);  printf("直接插入排序的结果为：\n");  for(i=1;i<=L.length;i++)  printf("%3d",L.r[i].key);  printf("\n");  break;  case 4: BubbleSort(L);  printf("冒泡排序的结果为：\n");  for(i=1;i<=L.length;i++)  printf("%3d",L.r[i].key);  printf("\n");  break;  case 5: quicksort(L,0,L.length);  printf("快速排序的结果为：\n");  for(i=1;i<=L.length;i++)  printf("%3d",L.r[i].key);  printf("\n");  break;  case 6: exit(0);  }  }  return 0;  } | | | |
| 6.实验结论及心得  1-表的初始化有点遗忘，回顾笔记  2-需要复习前面的内容，冒泡之类 | | | |