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： | 21130420 |
| 学 号： | 2113042015 |
| 姓 名： | 程楚晋 |
| 辅导教师： | 马巧梅 |

2022年03月制

成绩：

|  |  |  |  |
| --- | --- | --- | --- |
| 实验时间 | 2022年6月 8 日15 时至17 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：通过一趟排序将待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  // // Created by 86178 on 2022/6/3. // #include "stdio.h" #include<stdlib.h> #include "iostream" #define OK 1 #define ERROR 0 #define maxsize 20 const int N = 1e6+10; using namespace std; typedef struct{  int key;  float otherinfo; }rcdtype; typedef struct{  rcdtype r[maxsize + 1]; //r[0]闲置作为哨兵  int length; }sqlist; //插入排序算法 void insertsort(sqlist &L) {  int i,j;  for( i = 2; i <= L.length;i++ )  if (L.r[i].key < L.r[i-1].key)  { L.r[0]=L.r[i]; // L.r[0]为监视哨  for ( j=i-1; L.r[0].key < L.r[j].key; --j){  L.r[j+1]=L.r[j];}  L.r[j+1]=L.r[0];  } } //快排 int Partition(sqlist &L, int low, int high) {  L.r[0] = L.r[low];  int 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 Qsort(sqlist &L, int low, int high) {  if (low < high)  {  int pivotloc = Partition(L, low, high);  Qsort(L, low, pivotloc-1);  Qsort(L, pivotloc + 1, high);  } }  void Quicksort(sqlist &L) {  Qsort(L, 1, L.length); } //归并排序 void Merge(rcdtype R[],rcdtype T[],int low, int mid ,int high) { //将有序表归并  int i=low,j=mid+1,k=low;  while(i<=mid&&j<=high)  {  if(R[i].key<=R[j].key) T[k++]=R[i++];  else T[k++] = R[j++];  }  while(i<=mid) T[k++] = R[i++];  while(j<=mid) T[k++] = R[j++]; } void MSort(rcdtype R[],rcdtype T[],int low,int high) {  int mid;  rcdtype S[N];  if(low==high) T[low]=R[low];  else  {  mid = (low + high )/2;  MSort(R,S,low,mid);  MSort(R,S,mid+1,high);  Merge(S,T,low,mid,high);  } } void MergeSort(sqlist &L) {  MSort(L.r,L.r,1,L.length); } void Create\_Sq(sqlist &L) {  int n,i;  cout << "请输入数据个数，不超过" << maxsize << "个。" << endl;  cin >> n;  cout << ":请输入待排序的数据：\n";  while (n > maxsize )  {  cout << "个数上限，不能超过" << maxsize << "请重新输入：" << endl;  cin >> n;  }  for (i = 1; i <= n; i ++)  {  cin >> L.r[i].key;  L.r[i].otherinfo = i;  L.length ++;  } }  void show(sqlist L) {  int i;  for(i = 1;i <= L.length; i ++)  cout << L.r[i].key << " "; } //还原数据 void save(sqlist &L) {  int m = L.length - 1, flag = 1;  int j;  while ((m > 0) && flag == 1)  {  flag = 0; //本趟排序没有发生交换 ，将不会执行下一次  for (j = 1; j <= m; j ++)  {  if (L.r[j].otherinfo > L.r[j + 1].otherinfo)  {  flag = 1;  rcdtype t = L.r[j];  L.r[j] = L.r[j + 1];  L.r[j + 1] = t;  }  }  m --;  } }  int main() {  sqlist L;  L.length = 0;  Create\_Sq(L);  int i,j;  printf(" 排序 \n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("\* 1-----直接插入 \*\n");  printf("\* 2-----快速排序 \*\n");  printf("\* 3-----还原数据 \*\n");  printf("\* 4-----退出 \*\n");  printf("\* 5-----创建表 \*\n");  printf("\* 6-----归并排序 \*\n");  printf("\n");  while(1)  {  cout<<"请选择："<<endl;  cin>>j;  switch(j){  case 1:  insertsort(L);  cout<<"直接插入："<<endl;  show(L);  break;  case 2:  Quicksort(L);  cout<<"快速排序："<<endl;  show(L);  break;  case 3: save(L);  cout<<"还原数据："<<endl;  show(L);  break;  case 4:  exit (0);  case 5:  sqlist L1;  L1.length = 0;  Create\_Sq(L1);  case 6:  int x;  scanf("%d",&x);  // MSort(L,L1,1,x);  MergeSort(L1);  }  }  return 0; }  运行结果： | | | |
| 6.实验结论及心得  本次实验中，对排序有了新的认识,排序在很多领域都有广泛的应用。如各种升序学考试的录取工作，日常生活中的各类竞赛活动等都离不开排序，排序的一个主要目的是便于查找。不同排序有着不同的优点；比如选择排序会比较稳定，记录空间较小，当每一组记录占用的空间较多的时候，此方法比直接插入快，此外还有树形排序，归并排序，桶排序等等，其中，归并排序并没有掌握，后续会将实验补充完整。 | | | |