

天津商业大学学生实验报告

开课实验室：信息实验室

开课时间 13 年 9 月 1 日

实验报告 13 年 12 月 12 日

学院名称	信息工程	年级、专业、班	软件工程 12-01 班	学号	20125041	姓名	王靖伟	同组姓名	
课程名称	数据结构	实验项目名称	实验五 排序			指导教师	黄橡丽		
实验类型	验证 <input checked="" type="checkbox"/> 综合 <input type="checkbox"/> 设计 <input type="checkbox"/> 创新 <input type="checkbox"/>						成绩		
教师评语	教师签名：_____ 年 月 日								
实验报告内容一般包括以下几个内容：1、目的要求 2、仪器用具及材料（仪器名称及主要规格、用具名称） 3、实验内容及原理（简单但要抓住要点，写出依据原理） 4、操作方法与实验步骤 5、数据图表格（照片） 6、实验过程原始记录 7 数据处理及结果（按实验要求处理数据、结论） 8、作业题 9、讨论（对实验中存在的问题、进一步的想法等进行讨论）									
实验报告内容： 1. 实验目的、要求： （1）掌握各种排序（如：直接插入，希尔，冒泡，快速排序，简单选择，堆排序、等）方法及适用场合，并能在解决实际问题时灵活应用； （2）了解各种排序方法的排序过程及其依据的原理。 （3）学生用 C++/C 完成算法设计和程序设计并上机调试通过； （4）撰写实验报告，提供实验测试数据和实验结果； （5）分析算法，要求给出具体的算法分析结果，包括时间复杂度和空间复杂度，并简要给出算法设计小结和心得。 2. 实验内容(一)： （1）实验题目 编程实现直接插入排序算法。 （2）算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想） 1. 对顺序表 L 作直接插入排序 void insertsort(Sqlist &L) 2. 输出列表 void outlist(Sqlist L) 3. 主函数 void main() 当插入第 i 个记录时，前面的 R[1],R[2],...,R[i-1]已经排好序，这时采用 R[i]的关键字与 R[i-1], R[i-2],...的关键字依次进行比较，找到插入的位置，然后原位置上记录向后顺推，将 R[i]插入。这一过程称为一趟直接插入排序。 整个排序过程为 n-1 趟插入，即先将序列中第 1 个记录看成是一个有序子序列，然后从第 2 个记录开始，逐个进行插入，直至整个序列有序。 （3）程序清单									
行号	代码								
1	#include "stdio.h"								
2	#include "stdlib.h"								
3	#define MAXSIZE 10								

```

4  #define LT(a,b) ((a)<(b))
5  typedef int KeyType;
6  typedef struct{
7      KeyType key;
8      char name[20];
9  }RedType;
10 typedef struct{
11     RedType r[MAXSIZE+1];
12     int length;
13 }SqList;
14
15 void insertsort(SqList &L){
16     int i,j;
17     //请将该算法补充完整
18     // 对顺序表 L 作直接插入排序。
19     for (i=2; i<=L.length; ++i)
20         if (LT(L.r[i].key, L.r[i-1].key)) {
21             // "<"时, 需将 L.r[i]插入有序子表
22             L.r[0] = L.r[i];           // 复制为哨兵
23             for (j=i-1; LT(L.r[0].key, L.r[j].key); --j)
24                 L.r[j+1] = L.r[j];     // 记录后移
25             L.r[j+1] = L.r[0];         // 插入到正确位置
26         }
27 }
28
29 void outlist(SqList L)
30 {   int i;
31     for(i=1;i<=L.length;i++)
32         printf("%6d",L.r[i].key);
33     printf("\n");
34 }
35 void main()
36 {   int i;
37     SqList L;
38     printf("input length of list:");
39     scanf("%d",&L.length);
40     printf("input key:");
41     for(i=1;i<=L.length;i++)
42         scanf("%d",&L.r[i].key);
43     insertsort(L);
44     printf("result of sort:\n");
45     outlist(L);
46 }

```

(3) 测试数据

测试用例 1: 10 9 8 7 6 5 4 3 2 1

测试用例 1 的结果: 1 2 3 4 5 6 7 8 9 10

测试用例 2: 21 25 49 25 16 08
 测试用例 2 的结果: 08 16 21 25 25 49

(4) 实验结果

图 1.测试用例 1 的结果

图 2.测试用例 2 的结果

(5) 实验收获

设记录个数为 n ，则执行 $n-1$ 趟 比较次数和移动次数与初始排列有关
 最好情况下（关键字在记录序列中顺序有序）：
 每趟只需比较 1 次，移动 0 次
 总比较次数为 $n-1$
 时间复杂度为 $O(n^2)$
 空间复杂度为 $O(1)$
 是一种稳定的排序方法、算法简单，容易实现、也适用于链式结构（不需移动，只需修改指针）
 更适合初始记录基本有序（正序）的情况，也适合 n 较小的情况。当初始记录无序， n 较大时,不宜采用。

3. 实验内容(二):

(1) 实验题目

编程实现快速排序算法。

(2) 算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想）

1. 对顺序表 L 作快速排序排序 void quicksort(SqList & L ,int low,int high)
2. 返回枢轴所在位置 int partition(SqList & L ,int low,int high)
3. 输出列表 void outlist(SqList L)
4. 主函数 void main()

任取一个元素(如第一个) 为中心

所有比它小的元素一律前放，比它大的元素一律后放，形成左右两个子表；

对各子表重新选择中心元素并依此规则调整，直到每个子表的元素只剩一个

(3) 程序清单

行号	代码
1	#include "stdio.h"
2	#include "stdlib.h"

```

3  #define MAXSIZE 10
4  typedef int KeyType;
5  typedef struct{
6      KeyType key;
7      char name[20];
8  }RedType;
9  typedef struct{
10     RedType r[MAXSIZE+1];
11     int length;
12 }SqList;
13
14 int partition(SqList &L,int low,int high){
15     KeyType pivotkey;
16     RedType k;
17     pivotkey=L.r[low].key;
18
19     //请将该算法补充完整
20     RedType temp;
21     while (low<high) {          // 从表的两端交替地向中间扫描
22         while (low<high && L.r[high].key>=pivotkey) --high;
23         temp=L.r[low];
24         L.r[low]=L.r[high];
25         L.r[high]=temp;          // 将比枢轴记录小的记录交换到低端
26         while (low<high && L.r[low].key<=pivotkey) ++low;
27         temp=L.r[low];
28         L.r[low]=L.r[high];
29         L.r[high]=temp;          // 将比枢轴记录大的记录交换到高端
30     }
31     return low;                  // 返回枢轴所在位置
32 }
33
34 void quicksort(SqList &L,int low,int high){
35     int pivotpos;
36     if(low<high){
37         pivotpos=partition(L,low,high);
38         quicksort(L,low,pivotpos-1);
39         quicksort(L,pivotpos+1,high);
40     }
41 }
42 void outlist(SqList L)
43 {   int i;
44     for(i=1;i<=L.length;i++)
45         printf("%6d",L.r[i].key);
46     printf("\n");
47 }
48 void main()
49 {   int i;
50     SqList L;

```

```

51     printf("input length of list:");
52     scanf("%d",&L.length);
53     printf("input key:");
54     for(i=1;i<=L.length;i++)
55         scanf("%d",&L.r[i].key);
56     quicksort(L,1,L.length);
57     printf("result of sort:\n");
58     outlist(L);
59 }

```

(3) 测试数据

测试用例 1: 10 9 8 7 6 5 4 3 2 1

测试用例 1 的结果: 1 2 3 4 5 6 7 8 9 10

测试用例 2: 21 25 49 25 16 08

测试用例 2 的结果: 08 16 21 25 25 49

(4) 实验结果

图 3.测试用例 1 的结果

图 4.测试用例 2 的结果

(5) 实验收获

快速排序的记录移动次数不会大于比较次数，所以，快速排序的最坏时间复杂度为 $O(n^2)$ ；最好时间复杂度为 $O(n\log_2 n)$ 。可以证明，快速排序的平均时间复杂度也是 $O(n\log_2 n)$ 。

是一种不稳定的排序方法。

只适用于顺序结构

当 n 较大时，在平均情况下是所有内部排序方法中最快的一种，适合初始记录无序， n 较大的情况。

4. 实验内容(三):

(1) 实验题目

编程实现简单选择排序算法。

(2) 算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想）

1. 对顺序表 L 作简单选择排序 `void SelectSort(Sqlist &L)`

2. 输出列表 `void outlist(Sqlist L)`

3. 主函数 void main()

将待排序的记录分为已排序(初始为空)和未排序两组，依次将未排序的记录中值最小的结点放入已排序的组中的最后位置（未排序的组中的第一个位置）。

(3) 程序清单

行号	代码
1	#include "stdio.h"
2	#include "stdlib.h"
3	#define MAXSIZE 10
4	typedef int KeyType;
5	typedef struct{
6	KeyType key;
7	char name[20];
8	}RedType;
9	typedef struct{
10	RedType r[MAXSIZE+1];
11	int length;
12	}SqList;
13	
14	void SelectSort(SqList &L){
15	int i,j,k;
16	
17	//请将该算法补充完整
18	for (i=1; i<L.length; ++i) { // 选择第 i 小的记录，并交换到位
19	k=i;
20	for(j=i+1; j<=L.length ; j++)
21	if (L.r[j].key <L.r[k].key) k=j; // 在 L.r[i..L.length]中选择 key 最小的记录
22	if (i!=j) { // L.r[i] ↔ L.r[j]; 与第 i 个记录交换
23	RedType temp;
24	temp=L.r[i];
25	L.r[i]=L.r[k];
26	L.r[k]=temp;
27	}
28	}
29	}
30	
31	void outlist(SqList L)
32	{ int i;
33	for(i=1; i<=L.length; i++)
34	printf("%6d",L.r[i].key);
35	printf("\n");
36	}
37	void main()
38	{ int i;
39	SqList L;
40	printf("input length of list:");
41	scanf("%d",&L.length);

```

42     printf("input key:");
43     for(i=1;i<=L.length;i++)
44         scanf("%d",&L.r[i].key);
45     SelectSort(L);
46     printf("result of sort:\n");
47     outlist(L);
48 }

```

(3) 测试数据

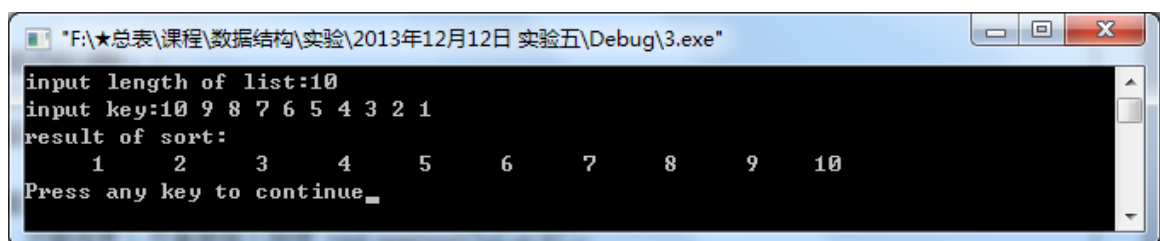
测试用例 1: 10 9 8 7 6 5 4 3 2 1

测试用例 1 的结果: 1 2 3 4 5 6 7 8 9 10

测试用例 2: 21 25 49 25 16 08

测试用例 2 的结果: 08 16 21 25 25 49

(4) 实验结果

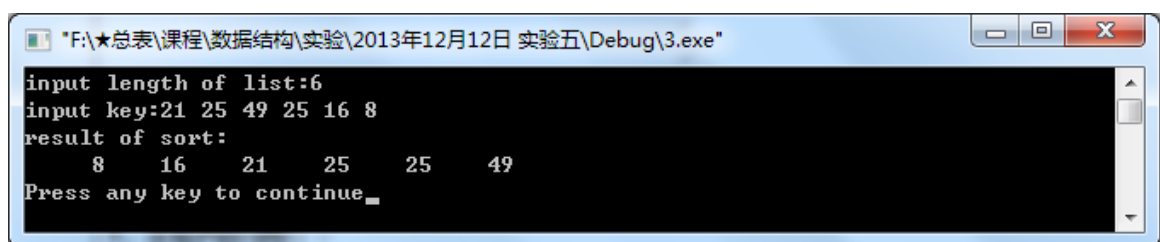


```

input length of list:10
input key:10 9 8 7 6 5 4 3 2 1
result of sort:
1      2      3      4      5      6      7      8      9      10
Press any key to continue_

```

图 5.测试用例 1 的结果



```

input length of list:6
input key:21 25 49 25 16 8
result of sort:
8      16      21      25      25      49
Press any key to continue_

```

图 6.测试用例 2 的结果

(5) 实验收获

时间复杂度

1.无论初始状态如何,都需进行 $n-1$ 趟排序, 在第 i 趟排序中选择最小关键字的记录, 需做 $n-i$ 次比较, 因此总的比较次数为: $O(n^2)$

2.当文件为正序时, 移动次数为 0, 文件初态为反序时,每趟排序均要执行交换操作,总的移动次数取最大值 $3(n-1)$ 。

空间复杂性为 $O(1)$ 。

直接选择排序是不稳定的排序方法。

5. 实验内容(四):

(1) 实验题目

编程实现堆排序算法。

(2) 算法设计思想 (说明整个程序由一个主函数和哪几个函数组成, 并给出主要函数的算法设计思想)

1. void heapify(SqList &L,int low,int high)

2.void buildheap(SqList &L)

3. 对记录序列 R[1..n]进行堆排序 void heapsort(SqList &L)
4. 输出列表 void outlist(SqList L)
5. 主函数 void main()

- (1) 将序列 r[1..n] 建初堆，交换 r[1]和 r[n]，则 r[n]为关键字最大的记录。
- (2) 将 r[1..n-1]重新调整为堆，交换 r[1]和 r[n-1]，则 r[n-1]为关键字次大的记录。
- (3) 循环 n-1 次，直到交换了 r[1]和 r[2]为止，得到了一个非递减的有序序列 r[1..n]。

(3) 程序清单

行号	代码
1	#include "stdio.h"
2	#include "stdlib.h"
3	#define MAXSIZE 10
4	#define LT(a,b) ((a)<(b))
5	typedef int KeyType;
6	typedef struct{
7	KeyType key;
8	char name[20];
9	}RedType;
10	typedef struct{
11	RedType r[MAXSIZE+1];
12	int length;
13	}SqList;
14	
15	void heapify(SqList &L,int low,int high)
16	{ int large;
17	RedType temp=L.r[low];
18	for(large=2*low;large<=high;large*=2)
19	{ if(large<high && L.r[large].key<L.r[large+1].key) large++;
20	if(temp.key>=L.r[large].key) break;
21	L.r[low]=L.r[large];
22	low=large;
23	}
24	L.r[low]=temp;
25	}
26	void buildheap(SqList &L)
27	{ int i,n=L.length;
28	for(i=n/2;i>0;i--)
29	heapify(L,i,n);
30	}
31	void heapsort(SqList &L)
32	{ int i,n=L.length;
33	buildheap(L);
34	for(i=n;i>1;i--){
35	L.r[0]=L.r[1];
36	L.r[1]=L.r[i];
37	L.r[i]=L.r[0];
38	heapify(L,1,i-1);


```

39     }
40 }
41
42 void outlist(SqList L)
43 {   int i;
44     for(i=1;i<=L.length;i++)
45         printf("%6d",L.r[i].key);
46     printf("\n");
47 }
48 void main()
49 {   int i;
49     SqList L;
50     printf("input length of list:");
51     scanf("%d",&L.length);
52     printf("input key:");
53     for(i=1;i<=L.length;i++)
54         scanf("%d",&L.r[i].key);
55     heapsort(L);
56     printf("result of sort:\n");
57     outlist(L);
58 }

```

(3) 测试数据

测试用例 1: 10 9 8 7 6 5 4 3 2 1

测试用例 1 的结果: 1 2 3 4 5 6 7 8 9 10

测试用例 2: 21 25 49 25 16 08

测试用例 2 的结果: 08 16 21 25 25 49

(4) 实验结果

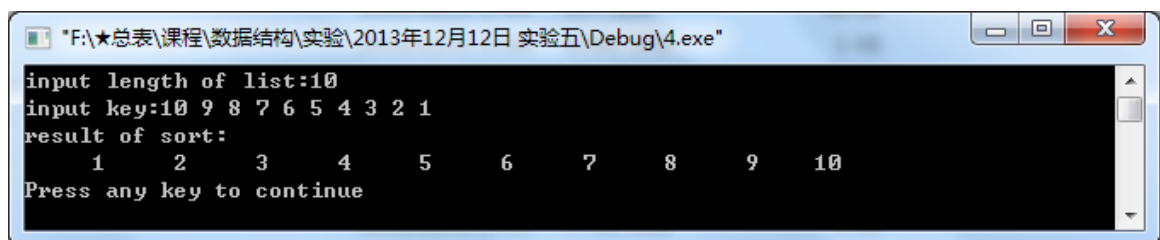


图 7.测试用例 1 的结果

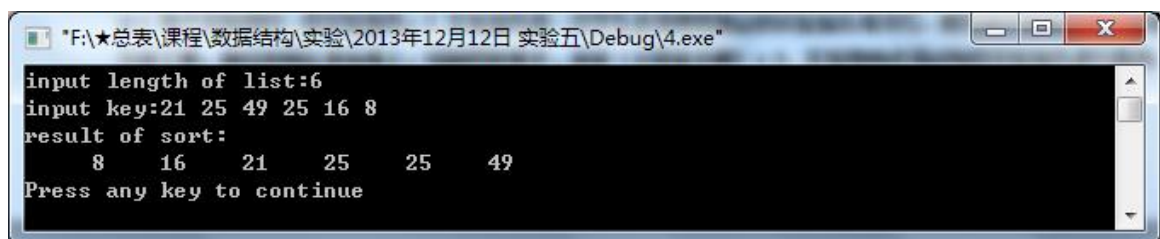


图 8.测试用例 2 的结果

(5) 实验收获

堆排序的时间复杂度为 $O(n\log n)$; 空间复杂度为 $O(1)$

稳定性: 不稳定

适用于 n 较大的情况

注 1. 每个实验项目一份实验报告。2. 实验报告第一页学生必须使用规定的实验报告纸书写, 附页用实验报告附页纸或 A4 纸书写, 字迹工整, 曲线要画在坐标纸上, 线路图要整齐、清楚 (不得徒手画)。3. 实验教师必须对每份实验报告进行批改, 用红笔指出实验报告中的错、漏之处, 并给出评语、成绩, 签全名、注明日期。4. 待实验课程结束以后, 要求学生把实验报告整理好, 交给实验指导教师, 加上实验课学生考勤及成绩登记表 (见附件 2)、目录和学院统一的封面 (见附件 3) 后, 统一装订成册存档。

制表单位: 设备处