**实验六 排序**

一、实验目的

1、掌握内部排序的基本算法；

2、分析比较内部排序算法的效率。

二、实验预习

说明以下概念

1、简单排序：

2、希尔排序：

3、快速排序：

4、堆排序：

三、实验内容和要求

1. 运行下面程序：

#include <stdlib.h>

#include <stdio.h>

#define MAX 50

int slist[MAX]; /\*待排序序列\*/

void insertSort(int list[], int n);

void createList(int list[], int \*n);

void printList(int list[], int n);

void heapAdjust(int list[], int u, int v);

void heapSort(int list[], int n);

/\*直接插入排序\*/

void insertSort(int list[], int n)

{

int i = 1, j = 0, node = 0, count = 1;

printf("对序列进行直接插入排序：\n");

printf("初始序列为：");

printList(list, n);

for(i = 2; i <= n; i++)

{

node = list[i];

j = i - 1;

while(j >= 0 && node < list[j])

{

list[j+1] = list[j];

--j;

}

list[j+1] = node;

printf("第%d次排序结果：", count++);

printList(list, n);

}

}

/\*堆排序\*/

void heapAdjust(int list[], int u, int v)

{

int i = u, j , temp = list[i];

j = 2 \* i;

while (j <= v)

{

if(j < v && list[j] < list[j+1])

j++; /\*若右孩子较大，则把j修改为右孩子的下标\*/

if(temp < list[j])

{

list[i] = list[j]; /\*将list[j]调到父亲的位置上\*/

i = j;

j = 2 \* i; /\*修改i和j的值，以便继续向下筛选\*/

}

else

break; /\*筛选完成，终止循环\*/

}

list[i] = temp; /\*被筛结点的值放入最终位置\*/

}

void heapSort(int list[], int n)

{

int i = 0, temp = 0, count = 1;

printf("对序列进行堆排序：\n");

printf("初始序列为：");

printList(list, n);

for (i = n / 2; i > 0; i--)

heapAdjust(list, i, n); /\*建立初始堆\*/

printf("建立的初始堆为：");

printList(list, n);

for(i = n ; i > 1; i--)

{/\*循环，完成堆排序\*/

temp = list[1];

list[1] = list[i];

list[i] = temp; /\*将第一个元素同当前区间内最后一个元素对换\*/

heapAdjust(list, 1 , i-1); /\*筛选出list[1]结点\*/

printf("第%d次排序结果：", count++);

printList(list, n);

}

}

/\*生成待排序序列\*/

void createList(int list[], int \*n)

{

int i = 1,a;

printf("请输入待排序序列(长度小于50,以输入一个字符结束):\n");

while(scanf("%d",&a)==1)

{

list[i] = a;

i++;

}

\*n=i-1;

getchar();

}

/\*输出排序结果\*/

void printList(int list[], int n)

{

int i = 1;

for(; i <= n; i++)

{

printf(" %d ", list[i]);

if(i % 10 ==0 && i != 0)

printf("\n");

}

printf("\n");

}

int main()

{

int choice=1,length;

while(choice!=0)

{

printf("\n");

printf("\*\*\*\*\* 内部排序算法演示程序 \*\*\*\*\*\n");

printf("\n1. 直接插入排序 \n");

printf("\n2. 堆排序 \n");

printf("\n0. 退出\n");

printf("\n请选择：");

scanf("%d", &choice);

getchar();

switch(choice)

{

case 1:

{

createList(slist, &length);

insertSort(slist, length);

break;

}

case 2:

{

createList(slist, &length);

heapSort(slist, length);

break;

}

default:choice=0;

}

printf("\n");

}

return 0;

}

输入待排序序列：49 38 65 97 13 27 49（以输入一个字符作为结束）

1）直接插入排序运行结果（写出每一趟的状态）：

2）堆排序运行结果（写出每一趟的状态）：

2、在1题中补充希尔排序算法。

算法代码：

输入待排序序列：49 38 65 97 13 27 49（以输入一个字符作为结束）

运行结果（写出每一趟的状态）：

3、在1题中补充快速排序算法。

算法代码：

输入待排序序列：49 38 65 97 13 27 49（以输入一个字符作为结束）

运行结果（写出每一趟的状态）：

四、实验小结

请比较各个排序算法的性能。