1.insersion sort

#include<stdio.h>

void insertion\_sort (int a[],int n)

{

int i,j,t;

for(i=1;i<n;i++)

{

j=1;

while(j>0&&a[j]<a[j-1])

{

t=a[j];

a[j]=a[j-1];

a[j-1]=t;

j=j-1;

j--;

}

}

}

int main()

{

int i,n,a[100];

printf("enter the number of elements");

scanf("%d",&n);

printf("enter the array elements");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

insertion\_sort(a,n);

printf("the sorted element in array\n");

for(i=0;i<n;i++)

printf("%d\t",a[i]);

}

Output: enter the number of elements5

enter the array elements2

3

6

7

8

the sorted element in array

2 3 6 7 8

--------------------------------

Process exited after 10.03 seconds with return value 0

Press any key to continue . . .

2.merge sort

#include <stdio.h>

void merge\_Sort(int a[], int low, int mid, int high)

{

int i, j, k, lo, temp[50];

lo = low;

i = low;

j = mid + 1;

while ((lo <= mid) && (j <= high))

{

if (a[lo] <= a[j])

{

temp[i] = a[lo];

lo++;

}

else

{

temp[i] = a[j];

j++;

}

i++;

}

if (lo > mid)

{

for (k = j; k <= high; k++)

{

temp[i] = a[k];

i++;

}

}

else

{

for (k = lo; k <= mid; k++)

{

temp[i] = a[k];

i++;

}

}

for (k = low; k <= high; k++)

a[k] = temp[k];

}

void partition(int a[],int low,int high)

{

int mid;

if(low < high)

{

mid = (low + high)/2;

partition( a, low, mid);

partition(a, mid+1, high);

merge\_Sort(a, low, mid, high);

}

}

int main()

{

int a[50] , i, n;

printf("Enter total number of elements:");

scanf("%d", &n);

printf("Enter the elements:\n");

for(i = 0; i < n; i++)

scanf("%d", &a[i]);

partition( a, 0, n - 1);

printf("After merge sort:\n");

for(i = 0;i < n; i++)

printf("%d\t", a[i]);

}

Output: Enter total number of elements:5

Enter the elements:

1

4

5

6

7

After merge sort:

1 4 5 6 7

--------------------------------

Process exited after 20.46 seconds with return value 0

Press any key to continue . . .

3.quick sort

#include<stdio.h>

void quicksort(int a[10],int first,int last)

{

int pivot,j,t,i;

if(first<last)

{

pivot=first;

i=first;

j=last;

while(i<j)

{

while(a[i] <= a[pivot])

i++;

while(a[j]>a[pivot])

j--;

if(i<j)

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

t=a[pivot];

a[pivot]=a[j];

a[j]=t;

quicksort(a,first,j-1);

quicksort(a,j+1,last);

}

}

int main()

{

int a[20],n,i;

printf("Enter size of the array: ");

scanf("%d",&n);

printf("Enter %d elements: ",n);

for(i=0;i<n;i++)

scanf("%d",&a[i]);

quicksort(a,0,n-1);

printf("Sorted elements: ");

for(i=0;i<n;i++)

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

return 0;

}

Output: Enter size of the array: 4

Enter 4 elements: 45

67

78

89

Sorted elements: 45 67 78 89

--------------------------------

Process exited after 8.718 seconds with return value 0

Press any key to continue . . .

4.heep sort

#include<stdio.h>

heapify(int a[], int n, int i)

{

int root,l,r,t;

root = i;

l = 2\*i + 1;

r = 2\*i + 2;

if (l<n && a[l] > a[root])

root = l;

if (r<n && a[r] > a[root])

root = r;

if (root != i)

{

t=a[i];

a[i] = a[root];

a[root] = t;

heapify(a, n, root);

}

}

heapsort(int a[], int n)

{

int i,t;

for(i=n/2-1; i>=0; i--)

heapify(a, n, i);

for(i=n-1; i>=0; i--)

{

t= a[0];

a[0]= a[i];

a[i] = t;

heapify(a, i, 0);

}

}

int main()

{

int a[50],i,n;

printf("Enter total number of elements:");

scanf("%d", &n);

printf("Enter the elements:\n");

for(i = 0; i < n; i++)

scanf("%d", &a[i]);

heapsort(a,n);

printf("\n\nAfter Heap sort:\n");

for(i = 0;i < n; i++)

printf("%d\t", a[i]);

return 0;

}

Output:

Enter total number of elements:4

Enter the elements:

2

7

9

4

After Heap sort:

2 4 7 9

--------------------------------

Process exited after 9.083 seconds with return value 0

Press any key to continue . . .