QUICK SORT

Experiment No.:5b

19/08/2012

# AIM:

# Implementation of Quick Sort on an array.

# ALGORITHM:

The following is the algorithm for quick sort on an array A of with upper bound ub and lower bound lb.

quick\_sort(A, lb, ub)

{

if(lb<ub)

{

p=partition(A, lb, ub); /\* defined following \*/

quick\_sort(A, lb, p-1);

quick\_sort(A, p+1, ub);

}

}

partition(A, lb, ub)

{

i=lb+1

j=ub

pivot=lb

while(i≤j)

{

if(A[i]<A[pivot])

{

Swap A[i] and A[pivot]

i=i+1;

pivot=pivot+1;

}

else

{

Swap A[i] and A[j]

j=j-1

}

}

return pivot;

}

# SOURCE CODE:

/\* Author : Rakesh Malik

\* Date : 11.08.2012

\* Quick Sort

\*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

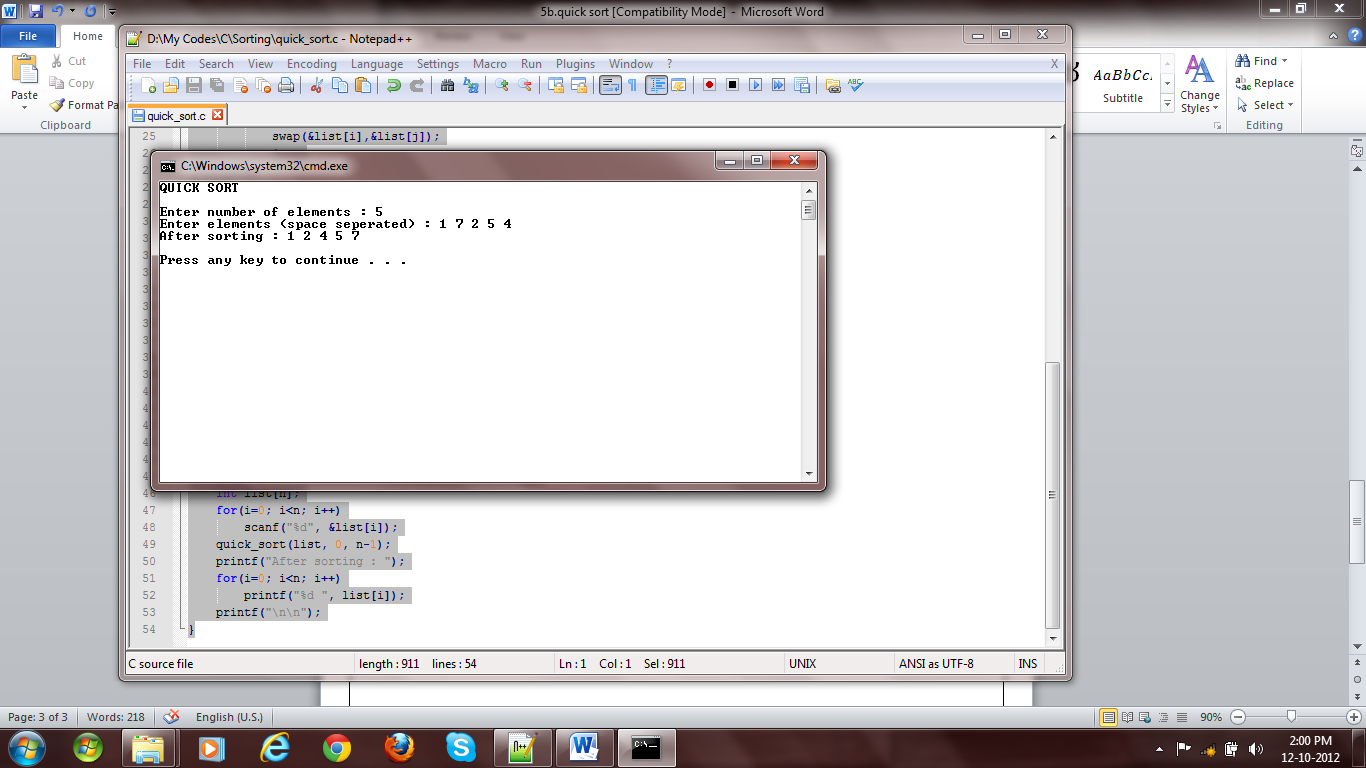


Fig 1: Quick Sort

void swap(int \*x,int \*y) {

int z=\*x;

\*x=\*y;

\*y=z;

}

int partition(int list[],int lb,int ub) {

int i=lb+1,j=ub,pivot=lb,temp;

while(i<=j) {

if(list[i]<list[pivot]) {

swap(&list[i],&list[pivot]);

i++;

pivot++;

} else {

swap(&list[i],&list[j]);

j--;

}

}

return pivot;

}

int quick\_sort(int list[], int lb, int ub) {

if(lb<ub) {

int p=partition(list, lb, ub);

quick\_sort(list, lb, p-1);

quick\_sort(list, p+1, ub);

}

}

int main() {

int n, i;

printf("QUICK SORT\n\n");

printf("Enter number of elements : ");

scanf("%d", &n);

printf("Enter elements (space seperated) : ");

int list[n];

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

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

quick\_sort(list, 0, n-1);

printf("After sorting : ");

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

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

printf("\n\n");

}

# RESULT:

The program has run successfully and given result as expected.