**PROGRAM-4**

**Aim:** Write an algorithm and program to sort n numbers using selection sort technique.

Algorithm :

**Step 1** − Set MIN to location 0

**Step 2** − Search the minimum element in the list

**Step 3** − Swap with value at location MIN

**Step 4** − Increment MIN to point to next element

**Step 5** − Repeat until list is sorted

**Source code:**

1. **USING ARRAY**

#include<stdio.h>

#include<conio.h>

void main(){

clrscr();

int min,t,k,a[25],n,i,j;

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

scanf("%d",&n);

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

for(k=0;k<n;k++){

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

}

printf("Array sorting:\n");

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

min=i;

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

if(a[min]>a[j]){

min=j;

}

}

if(min!=i){

t=a[min];

a[min]=a[i];

a[i]=t;

}

printf("\n");

for(k=0;k<n;k++){

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

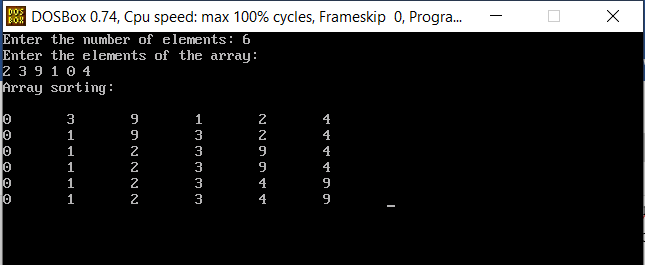
}

}

getch();

}

**OUTPUT:**



1. **USING RECURSION**

**Source code:**

#include<stdio.h>

#include<conio.h>

void sorting(int a[],int i,int n){

int j,min,k,t;

min=i;

if(i==n){

return;

}

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

if(a[min]>a[j]){

min=j;

}

}

if(min!=i){

t=a[min];

a[min]=a[i];

a[i]=t;

}

printf("\n");

for(k=0;k<n;k++){

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

}

printf("\n");

sorting(a,i+1,n);

}

void main(){

clrscr();

int a[25],n,k,i=0;

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

scanf("%d",&n);

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

for(k=0;k<n;k++){

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

}

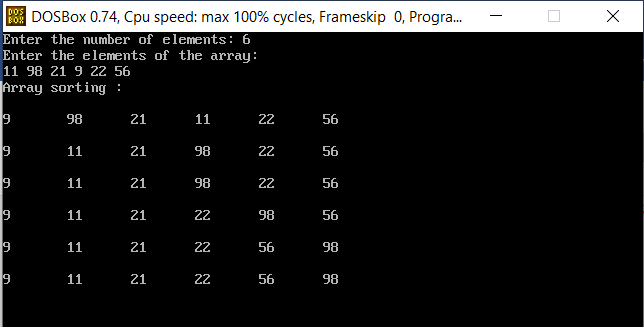
printf("Array sorting :\n");

sorting(a,i,n);

getch();

}

**OUTPUT:**



1. **USING RECURSION AND LINKED LIST**

**Source code:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct node{

int data;

struct node \*link;

};

struct node \*head,\*n;

void insert(){

int x;

struct node \*ptr;

ptr=head;

printf("Enter the value of new node: ");

scanf("%d",&x);

n=(struct node \*)malloc(sizeof(struct node));

n->data=x;

if(head==NULL){

n->link=NULL;

head=n;

}else{

while(ptr->link!=NULL){

ptr=ptr->link;

}

n->link=NULL;

ptr->link=n;

}

}

void traversal(){

struct node \*ptr;

ptr=head;

if(head==NULL){

printf("List is empty");

getch();

}

else{

while(ptr!=NULL){

printf("%d-> ",ptr->data);

ptr=ptr->link;

}

getch();

}

}

void sorted(int i,int n){

int j,t,k,w;

struct node \*temp,\*min,\*ptr;

w=i;

if(i==n){

return;

}

min=head;

temp=head;

while(w>1){

temp=temp->link;

min=min->link;

w--;

}

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

k=j;

ptr=head;

while(k>1){

ptr=ptr->link;

k--;

}

if(min->data>ptr->data){

min=ptr;

}

}

if(min!=temp){

t=min->data;

min->data=temp->data;

temp->data=t;

}

sorted(i+1,n);

}

void sorting(){

struct node \*ptr;

ptr=head;

int t=0;

if(head==NULL){

printf("List is empty");

getch();

}else{

while(ptr!=NULL){

ptr=ptr->link;

t=t+1;

}

sorted(1,t);

}

}

void main(){

int c;

L:system("cls");

printf("1. Insertion\n");

printf("2. Traversal\n");

printf("3. Sorting\n");

printf("4.Exit\n");

printf("Enter your choice: ");

scanf("%d",&c);

switch(c){

case 1:

insert();

goto L;

case 2:

traversal();

goto L;

case 3:

sorting();

getch();

goto L;

case 4:

exit(0);

default:

printf("Invalid choice...Enter your choice again");

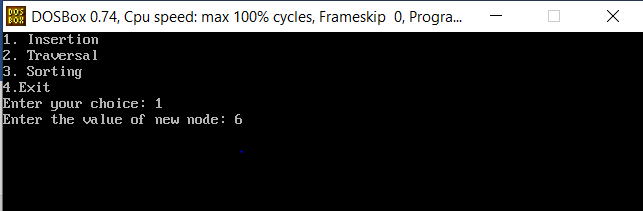
getch();

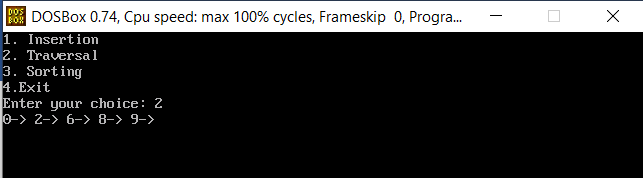
goto L;

}

}

**OUTPUT:**





**Complexity:**

Best case: O(n ˆ2)

Worst case: O(nˆ2)

Average case: O(nˆ2)