**Pimpri Chinchwad College of Engineering**

**Department of MCA**

**Data Structure Lab**

**Assignment No. 01 : Assignment based on Array Data Structure, Operations of Array Data Structure, Searching and Sorting, Application of Array**

**Date of Submission : 19th February 2021**

Write a menu driven program using C language to perform following operations on Array :

1. Insert
2. Delete
3. View
4. Update
5. Search

Search operation will have sub menus as : a) Linear Search b) Binary Search

**Solution :**

#include <stdio.h>

#include <stdlib.h>

#define MAX 50

int i,j,e,p,c,ne;

int main()

{

int n,arr[MAX];

printf("Enter the Number of Elements : ");

scanf("%d",&n);

printf("Enter the %d Elements : \n",n);

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

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

printf("\n Elements are : ");

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

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

while(1){

printf("\n\n ===== MAIN-MENU =====");

printf("\n 1.Insert");

printf("\n 2.Delete");

printf("\n 3.View");

printf("\n 4.Update");

printf("\n 5.Search");

printf("\n 6.Exit");

printf("\n\n Enter Your Choice : ");

scanf("%d",&c);

printf("================\n\n");

switch(c){

case 1 : n=InsertAt(arr,n);

break;

case 2 : n=DeleteAt(arr,n);

break;

case 3 : Display(arr,n);

break;

case 4 : n=UpdateAt(arr,n);

break;

case 5 : SearchAt(arr,n);

break;

case 6 : exit(0);

break;

default : printf("Please Enter Valid Choice");

break;

}

}

return 0;

}

void Display(int arr[MAX],int n){

printf("Elements are : ");

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

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

}

int InsertAt(int arr[MAX],int n){

printf("Enter the Element to be Insert : ");

scanf("%d",&e);

printf("Enter the Position between %d and %d : ",1,n+1);

scanf("%d",&p);

if(p>0 && p<=n+1){

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

arr[i] = arr[i-1];

arr[p-1] = e;

printf("\n Element Inserted!");

return ++n;

}else{

printf("\n Position Invalid!");

return n;

}

}

int DeleteAt(int arr[MAX],int n){

int cnt = 0;

while(1){

printf("\n ===== SUB-MENU =====");

printf("\n 1.Delete The Specific Element : ");

printf("\n 2.Delete the Element at Specific Position : ");

printf("\n 3.Back to Main Menu");

printf("\n 4.Exit");

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

scanf("%d",&c);

printf("================\n\n");

switch(c){

case 1 :printf("Enter the Element : ");

scanf("%d",&e);

int flag = 0;

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

if(arr[i]==e){

flag = 1;

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

arr[j] = arr[j+1];

}

cnt++;

printf("\n Element Deleted!");

}

}

if(flag==0)

printf("\n element %d not found",e);

n-=cnt;

return n;

break;

case 2 :printf("Enter the Postion between %d and %d : ",1,n);

scanf("%d",&p);

if(p>0 && p<=n){

if(n==p){

printf("Element Deleted!");

return --n;

}else{

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

arr[i-1] = arr[i];

}

printf("\n Element Deleted!");

return --n;

}

}else{

printf("\n Position Invalid!");

return n;

}

break;

case 3 : return 0;

break;

case 4 : exit(0);

break;

default : printf("Please Enter Valid Choice");

break;

}

}

}

int UpdateAt(int arr[MAX],int n){

int flag = 0;

int cnt = 0 ;

while(1){

printf("\n ===== SUB-MENU =====");

printf("\n 1.Update The Specific Element : ");

printf("\n 2.Update the Element at Specific Position : ");

printf("\n 3.Back to Main Menu");

printf("\n 4.Exit");

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

scanf("%d",&c);

printf("================\n\n");

switch(c){

case 1 :printf("Enter the Element to be update : ");

scanf("%d",&e);

printf("Enter the New Element to be replaced : ");

scanf("%d",&ne);

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

if(arr[i]==e){

arr[i] = ne;

cnt++;

flag = 1;

}

}

if(flag == 1)

printf("\n %d Element Updated !",cnt);

else

printf("\n Element Not Found");

return n;

break;

case 2 :printf("Enter the Postion between %d and %d : ",1,n);

scanf("%d",&p);

if(p>0 && p<=n){

printf("Enter the New Element to be replaced : ");

scanf("%d",&ne);

arr[p-1] = ne;

printf("\n Element Updated!");

return n;

}else{

printf("Position Invalid!");

return n;

}

break;

case 3 : return 0;

break;

case 4 : exit(0);

break;

default : printf("Please Enter Valid Choice");

break;

}

}

}

void SearchAt(int arr[MAX],int n){

while(1){

printf("\n\n===== SUB-MENU =====");

printf("\n 1.Linear Search : ");

printf("\n 2.Binary Search : ");

printf("\n 3.Back to Main Menu");

printf("\n 4.Exit");

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

scanf("%d",&c);

printf("================\n\n");

switch(c){

case 1 : LinearSearch(arr,n);

break;

case 2 :printf("Enter the Element to be search : ");

scanf("%d",&e);

int result = binarySearch(arr,0,n-1,e);

if(result == -1)

printf("\n Element is not Found");

else

printf("\n Element is found at %d index",result);

break;

case 3 : return 0;

break;

case 4 : exit(0);

break;

default : printf("Please Enter Valid Choice");

break;

}

}

}

void LinearSearch(int arr[],int n){

printf("Enter the Element to be search : ");

scanf("%d",&e);

int flag = 0;

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

if(arr[i]==e){

flag = 1;

printf("\n element %d found at index %d",e,i);

}

}

if(flag == 0){

printf("\n Element Not Found");

}

}

int binarySearch(int arr[],int f,int l,int e){

int temp;

for(i=0;i<l+1;i++){

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

if(arr[i]>arr[j]){

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

while(f<=l){

int m = f + (l-f)/2;

if(arr[m]==e)

return m;

if(arr[m]<e)

f = m+1;

else

l = m-1;

}

return -1;

}

void sort(int arr[MAX],int n){

int temp;

while(1){

printf("\n\n ===== SUB-MENU =====");

printf("\n 1.Ascending : ");

printf("\n 2.Descending : ");

printf("\n 3.Back to Main Menu");

printf("\n 4.Exit");

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

scanf("%d",&c);

printf("================\n\n");

switch(c){

case 1 : ascending(arr,n);

break;

case 2 : descending(arr,n);

break;

case 3 : return 0;

break;

case 4 : exit(0);

break;

default : printf("Please Enter Valid Choice");

break;

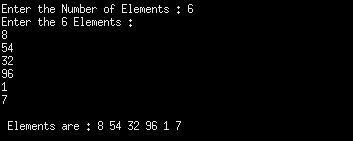
}

}

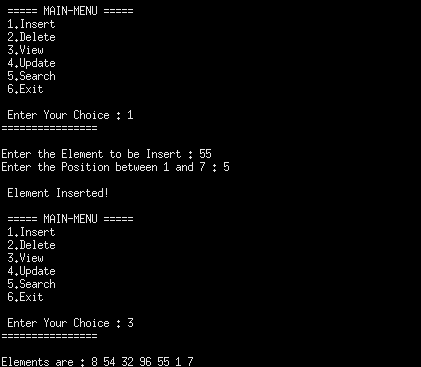
}

***Output***

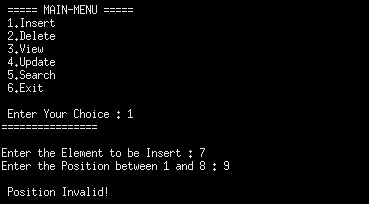
0 . Accept the input:



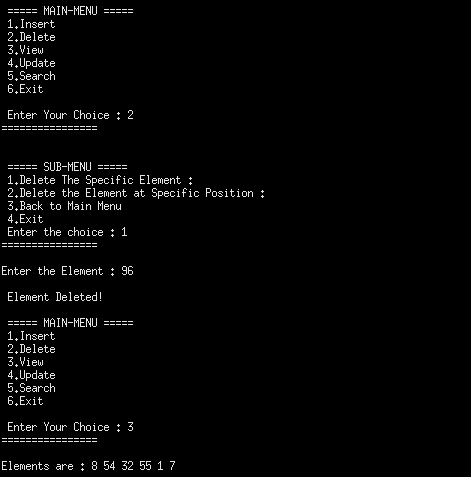
1 . Insert the Element :



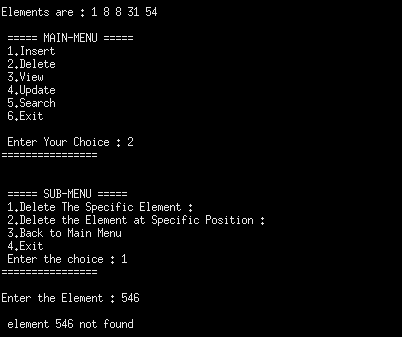
1 validation for inserting element :



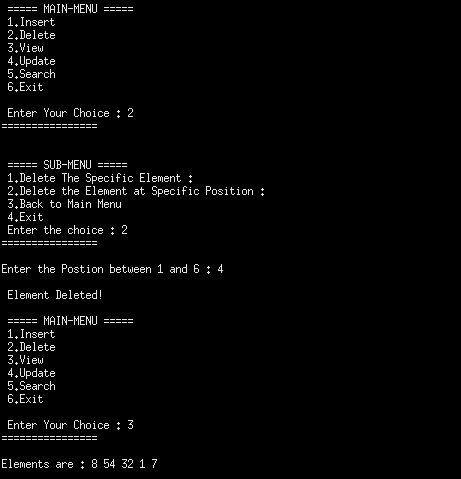
2.1 Delete Specific Element :



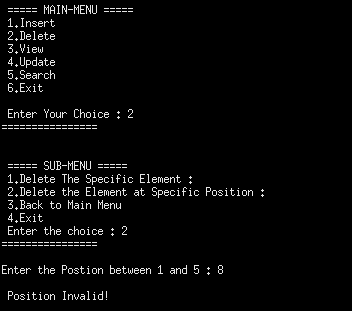
2.1 Validation for Delete Specific Element :



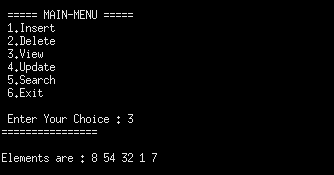
2.2 Delete Element at Specific Position :

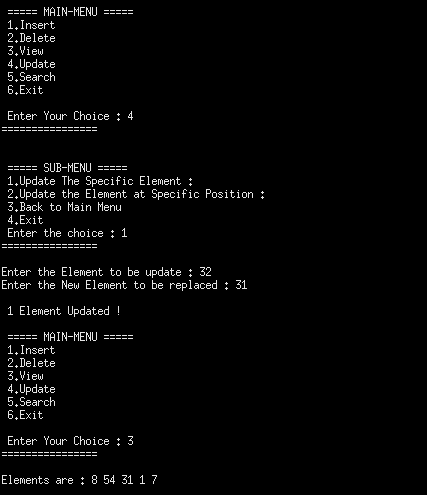


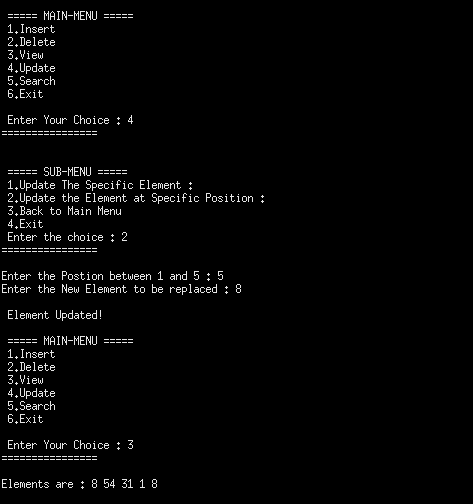
2.2 Validation for Delete Element at Specific Postion :



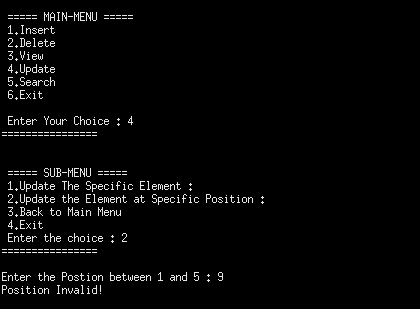
3. View the Elements of Array :



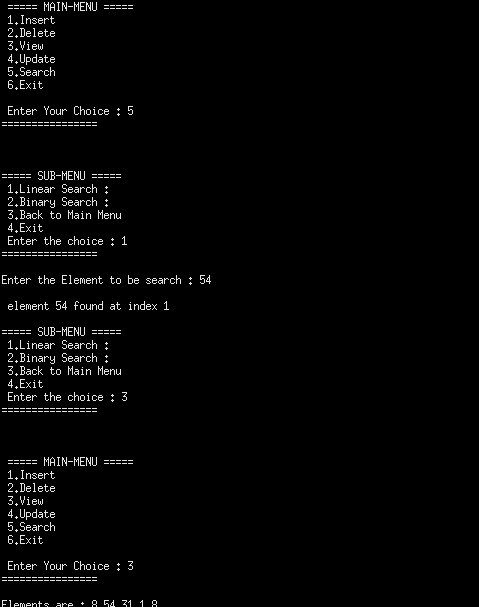
4.1 Update Specific Element :

4.2 Update Element at Specific Postion :

4.2 Validation for Update Element at Specific Postion :



5.1 Linear Search :



5.2 Binary Search :

