1. **Stack using arrays**

#include<iostream>

using namespace std;

#define MAX 5

int TOP,status;

void push(int stack[],int item)

{

if(TOP==(MAX-1))

status=0;

else

{

status=1;

++TOP;

stack[TOP]=item;

}

}

int pop(int stack[])

{

int ret;

if(TOP==-1)

{

ret=0;

status=0;

}

else

{

status=1;

ret=stack[TOP];

--TOP;

}

return ret;

}

void display(int stack[])

{

int i;

cout<<"\nThe stack is:"<<endl;

if(TOP==-1)

cout<<"Stack is empty"<<endl;

else

{

for(i=TOP;i>=0;i--)

{

cout<<stack[i]<<"\t";

}

cout<<endl;

}

cout<<endl;

}

int main()

{

int ch;

int stack[MAX],item;

TOP=-1;

do

{

do

{

cout<<"MAIN MENU"<<endl;

cout<<"1.PUSH"<<endl;

cout<<"2.POP"<<endl;

cout<<"3.EXIT"<<endl;

cout<<"Enter your choice: ";

cin>>ch;

if(ch<1 || ch>3)

cout<<"Invalid choice, try again!"<<endl;

}while(ch<1 || ch>3);

switch(ch)

{

case 1:

cout<<"Enter the element to be pushed: ";

cin>>item;

cout<<"Item="<<item<<endl;

push(stack,item);

if(status)

{

cout<<endl<<"After pushing";

display(stack);

if(TOP==(MAX-1))

cout<<"Stack is full";

}

else

cout<<"Stack is overflow";

break;

case 2:

item=pop(stack);

if(status)

{

cout<<endl<<"The popped item= "<<item;

display(stack);

}

else

cout<<endl<<"Stack is underflow"<<endl;

break;

default:

cout<<endl<<"END OF PROGRAM"<<endl;

}

}while(ch!=3);

}

**OUTPUT:**

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 1

Enter the element to be pushed: 3

Item=3

After pushing

The stack is:

3

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 1

Enter the element to be pushed: 5

Item=5

After pushing

The stack is:

5 3

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 2

The popped item= 5

The stack is:

3

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 3

END OF PROGRAM

1. **Stack using linked list**

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node \*next;

}\*TOP;

void push(int item)

{

struct node \*ptr;

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

ptr->data=item;

if(TOP==NULL)

{

TOP=ptr;

TOP->next=NULL;

}

else

{

ptr->next=TOP;

TOP=ptr;

}

}

void pop()

{

struct node \*ptr;

ptr=TOP;

if(TOP==NULL)

{

cout<<"Stack is empty!!"<<endl;

}

else

{

TOP=TOP->next;

free(ptr);

}

}

void display()

{

int i;

struct node \*ptr;

ptr=TOP;

if(TOP==NULL)

{

cout<<"Stack is empty!!"<<endl;

}

else

{

cout<<"Stack elements are: ";

while(ptr!=NULL)

{

cout<<ptr->data<<"\t";

ptr=ptr->next;

}

cout<<endl;

}

}

int main()

{

int item,ch;

TOP=NULL;

cout<<"1.PUSH\n";

cout<<"2.POP\n";

cout<<"3.DISPLAY\n";

cout<<"4.EXIT\n";

while(1)

{

cout<<"Enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

{

cout<<"Enter element to be pushed: ";

cin>>item;

push(item);

display();

break;

}

case 2:

{

pop();

cout<<"After popping"<<endl;

display();

break;

}

case 3:

{

display();

break;

}

case 4:

{

exit(0);

break;

}

}

}

}

**OUTPUT:**

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to be pushed: 4

Stack elements are: 4

Enter your choice: 1

Enter element to be pushed: 7

Stack elements are: 7 4

Enter your choice: 2

After popping

Stack elements are: 4

Enter your choice: 3

Stack elements are: 4

Enter your choice: 4

1. **Queue using arrays**

#include<iostream>

using namespace std;

#include<stdlib.h>

#define MAX 5

int queue[MAX];

int front=-1;

int rear=-1;

void insert()

{

int element;

if(rear==(MAX-1))

{

cout<<"Queue full!"<<endl;

}

else

{

if(front==-1)

{ front=0; }

cout<<"Enter element: ";

cin>>element;

queue[++rear]=element;

}

}

void remove()

{

int element;

if(front==-1)

cout<<"Queue is empty!"<<endl;

else

{

cout<<"Element "<<queue[front]<<" is deleted!\n\n";

++front;

if(front>rear)

{

front=-1;

rear=-1;

}

}

}

void display()

{

int i;

if(front==-1)

cout<<"Queue is empty!\n";

else

{

cout<<"Queue elements are: ";

for(i=front;i<=rear;i++)

cout<<queue[i]<<"\t";

}

cout<<endl<<endl;

}

int main()

{

int ch;

while(1)

{

cout<<"1.Insert"<<endl;

cout<<"2.Delete"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Exit"<<endl;

cout<<"\nEnter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

{

insert();

display();

} break;

case 2:

{

remove();

display();

break;

}

case 3:

{

display();

break;

}

case 4:

{

exit(0);

break;

}

default:

cout<<"Invalid choice!"<<endl;

}

}

}

**OUTPUT:**

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 1

Enter element: 8

Queue elements are: 8

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 1

Enter element: 2

Queue elements are: 8 2

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 2

Element 8 is deleted!

Queue elements are: 2

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 4

1. **Queue using linked list**

#include<iostream>

using namespace std;

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*rear,\*front;

typedef struct node NODE;

void insert(int value)

{

NODE \*temp;

temp=(NODE\*)malloc(sizeof(NODE));

temp->data=value;

if(front==NULL)

{

front=temp;

front->next=NULL;

rear=front;

}

else

{

rear->next=temp;

rear=temp;

rear->next=NULL;

}

}

void remove()

{

NODE \*temp;

temp=front;

if(front==NULL)

{

cout<<"EMPTY!!"<<endl;

}

else

{

temp=front;

front=front->next;

free(temp);

}

}

void display()

{

NODE \*temp;

temp=front;

if(front==NULL)

{

cout<<"EMPTY!!"<<endl;

}

else if(temp!=NULL)

{

cout<<"Elements are: ";

while(temp!=NULL)

{

cout<<temp->data;

temp=temp->next;

}

cout<<endl;

}

}

int main()

{

int i=0;

front=NULL;

cout<<"1.Insert"<<endl;

cout<<"2.Delete"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Exit"<<endl;

while(1)

{

cout<<"Enter choice: ";

cin>>i;

switch(i)

{

case 1:

int value;

cout<<"Enter data: ";

cin>>value;

insert(value);

display();

break;

case 2:

remove();

display();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

cout<<"Invalid choice!!"<<endl;

}

}

}

**OUTPUT:**

1.Insert

2.Delete

3.Display

4.Exit

Enter choice: 1

Enter data: 8

Elements are: 8

Enter choice: 1

Enter data: 2

Elements are: 8 2

Enter choice: 2

Elements are: 2

Enter choice: 3

Elements are: 2

Enter choice: 4

1. **Singly linked list**

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int num;

struct node \*next;

};

typedef struct node NODE;

NODE \*start,\*news;

void insert\_at\_beg(int item)

{

NODE \*news;

news=(NODE\*)malloc(sizeof(NODE));

news->num=item;

if(start==NULL)

news->next=NULL;

else

news->next=start;

start=news;

}

void insert\_at\_anywhere(int value,int loc)

{

struct node \*var,\*temp;

news=(NODE\*)malloc(sizeof(NODE));

news->num=value;

temp=start;

if(start==NULL)

{

start=news;

start->next=NULL;

}

else

while(temp->num!=loc)

{

temp=temp->next;

}

var=temp->next;

temp->next=news;

news->next=var;

}

void insert\_at\_end(int item)

{

NODE \*news,\*temp;

news=(NODE\*)malloc(sizeof(NODE));

news->num=item;

news->next=NULL;

if(start==NULL)

start=news;

else

temp=start;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=news;

}

void remove()

{

struct node \*temp;

if(start==NULL)

cout<<"List is empty!\n";

else

{

temp=start;

start=start->next;

free(temp);

cout<<"\nElement deleted successfully!\n";

}

}

void display()

{

struct node \*temp;

if(start==NULL)

{

cout<<"\nList is empty!\n";

return;

}

temp=start;

cout<<"List of elements:\n";

while(temp!=NULL)

{

cout<<temp->num<<" ";

temp=temp->next;

}

cout<<endl;

}

int main()

{

int ch,item;

do

{

cout<<"\n 1.insert\_at\_beg";

cout<<"\n 2.insert\_anywhere";

cout<<"\n 3.insert\_at\_end";

cout<<"\n 4.delete";

cout<<"\n 5.display";

cout<<"\n 6.exit";

cout<<"\n Enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

{

cout<<"\nEnter element: \n";

cin>>item;

insert\_at\_beg(item);

display();

break;

}

case 2:

{

int value,loc;

cout<<"\n After which data item you want to insert: ";

cin>>loc;

cout<<"\n Enter value to insert: ";

cin>>value;

insert\_at\_anywhere(value,loc);

display();

break;

}

case 3:

{

cout<<"\nEnter element: ";

cin>>item;

insert\_at\_end(item);

display();

break;

}

case 4:

{

remove();

display();

break;

}

case 5:

display();

break;

case 6:

exit(0);

}

}

while(ch!=6);

}

**OUTPUT:**

1.insert\_at\_beg

2.insert\_anywhere

3.insert\_at\_end

4.delete

5.display

6.exit

Enter your choice: 1

Enter element:

12

List of elements:

12

1.insert\_at\_beg

2.insert\_anywhere

3.insert\_at\_end

4.delete

5.display

6.exit

Enter your choice: 3

Enter element: 18

List of elements:

12 18

1.insert\_at\_beg

2.insert\_anywhere

3.insert\_at\_end

4.delete

5.display

6.exit

Enter your choice: 2

After which data item you want to insert: 12

Enter value to insert: 16

List of elements:

12 16 18

1.insert\_at\_beg

2.insert\_anywhere

3.insert\_at\_end

4.delete

5.display

6.exit

Enter your choice: 4

Element deleted successfully!

List of elements:

16 18

1.insert\_at\_beg

2.insert\_anywhere

3.insert\_at\_end

4.delete

5.display

6.exit

Enter your choice: 6

1. **Doubly liked list**

#include<iostream>

using namespace std;

#include<stdlib.h>

struct node

{

struct node \*previous;

int data;

struct node \*next;

}\*start,\*last;

void insert\_beg(int value)

{

struct node \*ptr,\*temp;

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

ptr->data=value;

if(start==NULL)

{

start=ptr;

start->previous=NULL;

start->next=NULL;

last=start;

}

else

{

ptr->previous=NULL;

ptr->next=start;

start->previous=ptr;

start=ptr;

}

}

void insert\_end(int value)

{

struct node \*ptr,\*temp;

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

ptr->data=value;

if(start==NULL)

{

start=ptr;

start->previous=NULL;

start->next=NULL;

last=start;

}

else

{

last=start;

while(last!=NULL)

{

temp=last;

last=last->next;

}

last=ptr;

temp->next=last;

last->previous=temp;

last->next=NULL;

}

}

void insert\_after(int value,int loc)

{

struct node \*temp,\*ptr,\*temp1;

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

ptr->data=value;

if(start==NULL)

{

start=ptr;

start->previous=NULL;

start->next=NULL;

}

{

temp=start;

while((temp!=NULL)&&(temp->data)!=loc)

{

temp=temp->next;

}

if(temp==NULL)

{

cout<<loc<<" not present in the list";

}

else

{

temp1=temp->next;

temp->next=ptr;

ptr->previous=temp;

ptr->next=temp1;

temp1->previous=ptr;

}

}

last=start;

while(last->next!=NULL)

{

last=last->next;

}

}

void delet()

{

struct node \*temp;

temp=start;

start=start->next;

free(temp);

cout<<"\nElement deleted successfully";

}

void display()

{

struct node \*temp;

temp=start;

if(temp==NULL)

{

cout<<"list is empty"<<endl;

}

while(temp!=NULL)

{

cout<<temp->data;

temp=temp->next;

}

}

int main()

{

int value,i,loc;

start=NULL;

cout<<"select your choice:"<<endl;

cout<<"\n1.insert at beginning";

cout<<"\n2.insert at end";

cout<<"\n3.insert anywhere";

cout<<"\n4.delete";

cout<<"\n5.display";

cout<<"\n6.exit";

while(1)

{

cout<<"\n\nEnter your choice:";

cin>>i;

switch(i)

{

case 1:

{

cout<<"\nEnter data to insert:";

cin>>value;

insert\_beg(value);

display();

break;

}

case 2:

{

cout<<"\nEnter value:";

cin>>value;

insert\_end(value);

display();

break;

}

case 3:

{

cout<<"\nAfter which data item you want to insert:";

cin>>loc;

cout<<"\nenter data to insert";

cin>>value;

insert\_after(value,loc);

display();

break;

}

case 4:

{

delet();

display();

break;

}

case 5:

{

display();

break;

}

case 6:

exit(0);

break;

}

}

}

**OUTPUT:**

select your choice:

1.insert at beginning

2.insert at end

3.insert anywhere

4.delete

5.display

6.exit

Enter your choice:1

Enter data to insert:8

8

Enter your choice:1

Enter data to insert:2

28

Enter your choice:2

Enter value:7

287

Enter your choice:3

After which data item you want to insert:8

enter data to insert6

2867

Enter your choice:4

Element deleted successfully867

Enter your choice:6

1. **Tree Traversal**

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

char data;

struct node \*left, \*right;

};

typedef struct node NODE;

NODE\* makeNode(char n)

{

NODE \*newn;

newn = (NODE\*)malloc(sizeof(NODE));

newn -> data = n;

newn -> left = NULL;

newn -> right = NULL;

return newn;

}

void preOrder(NODE \*temp)

{

if(temp != NULL)

{

cout << temp -> data << endl;

preOrder(temp -> left);

preOrder(temp -> right);

}

}

void inOrder(NODE \*temp)

{

if(temp != NULL)

{

inOrder(temp -> left);

cout << temp -> data << endl;

inOrder(temp -> right);

}

}

void postOrder(NODE \*temp)

{

if(temp != NULL)

{

postOrder(temp -> left);

postOrder(temp -> right);

cout << temp -> data << endl;

}

}

int main()

{

NODE \*tree;

tree = makeNode('A');

tree -> left = makeNode('B');

tree -> right = makeNode('C');

cout << "PreOrder: " << endl;

preOrder(tree);

cout << "InOrder: " << endl;

inOrder(tree);

cout << "PostOrder: " << endl;

postOrder(tree);

}

**OUTPUT:**

PreOrder:

A

B

C

InOrder:

B

A

C

PostOrder:

B

C

A

1. **Selection Sort**

#include<iostream>

using namespace std;

int main()

{

int n, i, a[50], j, temp;

cout<<"Enter total number of elements:";

cin>>n;

cout<<"Enter numbers:";

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

{

cin>>a[i];

}

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

{

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

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

cout<<"Sorted list:"<<endl;

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

{

cout<<a[i]<<" ";

}

}

**OUTPUT:**

Enter total number of elements:5

Enter numbers:42 10 2 33 5

Sorted list:

2 5 10 33 42

1. **Bubble Sort**

#include<iostream>

using namespace std;

int main()

{

int n, i, a[50], j, temp;

cout<<"Enter total number of elements:";

cin>>n;

cout<<"Enter numbers:";

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

{

cin>>a[i];

}

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

{

for(j=0; j<(n-i-1); j++)

{

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

cout<<"Sorted list:"<<endl;

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

{

cout<<a[i]<<" ";

}

}

**OUTPUT:**

Enter total number of elements:6

Enter numbers:8 3 9 4 1 7

Sorted list:

1 3 4 7 8 9

1. **Merge Sort**

#include <iostream>

using namespace std;

void merge(int a[], int p, int q, int r)

{

int b[5];

int i, j, k;

k = 0;

i = p;

j = q + 1;

while(i <= q && j <= r)

{

if(a[i] < a[j])

{

b[k++] = a[i++];

}

else

{

b[k++] = a[j++];

}

}

while(i <= q)

{

b[k++] = a[i++];

}

while(j <= r)

{

b[k++] = a[j++];

}

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

{

a[i] = b[--k];

}

}

void mergeSort(int a[], int p, int r)

{

int q;

if(p < r)

{

q = (p + r) / 2;

mergeSort(a, p, q);

mergeSort(a, q+1, r);

merge(a, p, q, r);

}

}

void printArray(int a[], int size)

{

int i;

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

{

cout<<a[i]<<" ";

}

cout<<"\n";

}

int main()

{

int arr[50],n,i;

cout<<"Enter size of array: ";

cin>>n;

cout<<"Enter Elements:\n";

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

{ cin>>arr[i]; }

mergeSort(arr, 0, n - 1);

cout<<"Sorted array: \n";

printArray(arr, n);

return 0;

}

**OUTPUT:**

Enter size of array: 5

Enter Elements:

9 7 5 3 1

Sorted array:

1 3 5 7 9

1. **Quick Sort**

#include<iostream>

using namespace std;

void quicksort(int number[25],int first,int last)

{

int i, j, pivot, temp;

if(first<last)

{

pivot=first;

i=first;

j=last;

while(i<j)

{

while(number[i]<=number[pivot]&&i<last)

i++;

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

j--;

if(i<j)

{

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

temp=number[pivot];

number[pivot]=number[j];

number[j]=temp;

quicksort(number,first,j-1);

quicksort(number,j+1,last);

}

}

int main()

{

int i, count, number[25];

cout<<"Enter number of elements: ";

cin>>count;

cout<<"Enter elements:\n";

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

cin>>number[i];

quicksort(number,0,count-1);

cout<<"The Sorted Order is:\n";

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

cout<<number[i]<<" ";

cout<<endl;

return 0;

}

**OUTPUT:**

Enter number of elements: 6

Enter elements:

8 2 4 0 1 7

The Sorted Order is:

0 1 2 4 7 8

1. **DFS**

#include<iostream>

using namespace std;

int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];

int main()

{

int m;

cout <<"Enter no of vertices:";

cin >> n;

cout <<"Enter no of edges:";

cin >> m;

cout <<"\nEDGES \n";

for(k=1; k<=m; k++)

{

cin >>i>>j;

cost[i][j]=1;

}

cout <<"Enter initial vertex to traverse from:";

cin >>v;

cout <<"DFS ORDER OF VISITED VERTICES:";

cout << v <<" ";

visited[v]=1;

k=1;

while(k<n)

{

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

if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)

{

visit[j]=1;

stk[top]=j;

top++;

}

v=stk[--top];

cout<<v << " ";

k++;

visit[v]=0;

visited[v]=1;

}

return 0;

}

**OUTPUT:**

Enter no of vertices:7

Enter no of edges:7

EDGES

1 2

1 3

2 4

2 5

2 6

3 6

3 7

Enter initial vertex to traverse from:1

DFS ORDER OF VISITED VERTICES:1 2 4 5 6 3 7

1. **BFS**

#include<iostream>

using namespace std;

int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10];

int main()

{

int m;

cout <<"Enter number of vertices: ";

cin >> n;

cout <<"Enter number of edges: ";

cin >> m;

cout <<"\nEDGES\n";

for(k=1; k<=m; k++)

{

cin >>i>>j;

cost[i][j]=1;

}

cout <<"Enter initial vertex to traverse from: ";

cin >>v;

cout <<"BFS order of visited vertices: ";

cout <<v<<" ";

visited[v]=1;

k=1;

while(k<n)

{

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

if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)

{

visit[j]=1;

qu[rare++]=j;

}

v=qu[front++];

cout<<v <<" ";

k++;

visit[v]=0;

visited[v]=1;

}

return 0;

}

**OUTPUT:**

Enter number of vertices: 7

Enter number of edges: 7

EDGES

1 2

1 3

2 4

2 5

2 6

3 6

3 7

Enter initial vertex to traverse from: 1

BFS order of visited vertices: 1 2 3 4 5 6 7

1. **Linear Search**

#include<iostream>

using namespace std;

int e[500],k,n;

int linearsearch(int e[],int n,int k)

{

int i;

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

{

if(k==e[i])

{

return i;

break;

}

}

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

{

if(k!=e[i])

{ return -1; }

}

}

int main()

{

int i,pos;

cout<<"ENTER NUMBER OF ELEMENTS IN THE LIST: ";

cin>>n;

cout<<"ENTER THE ELEMENTS ONE BY ONE: "<<endl;

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

{

cin>>e[i];

}

cout<<"ENTER THE KEY TO BE SEARCHED: ";

cin>>k;

pos=-1;

pos=linearsearch(e,n,k);

if(pos!=-1)

{ cout<<"ELEMENT IS IN THE POSITION - "<<pos+1<<endl; }

else

{ cout<<"ELEMENT IS NOT FOUND IN THE LIST!"<<endl; }

}

**OUTPUT:**

ENTER NUMBER OF ELEMENTS IN THE LIST: 6

ENTER THE ELEMENTS ONE BY ONE:

3 9 4 2 0 7

ENTER THE KEY TO BE SEARCHED: 2

ELEMENT IS IN THE POSITION – 4

1. **Binary Search**

#include<iostream>

using namespace std;

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

void binarysearch(int element,int n,int list[]);

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

int main()

{

int n,i,key,list[100];

cout<<"ENTER NUMBER OF ELEMENTS IN THE LIST: ";

cin>>n;

cout<<"ENTER THE ELEMENTS ONE BY ONE: "<<endl;

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

{

cin>>list[i];

}

cout<<"ENTER THE KEY TO BE SEARCHED: ";

cin>>key;

sort(list,n);

output(list,n);

binarysearch(key,n,list);

}

void sort(int list[],int n)

{

int t;

for(int j=0;j<n;j++)

{

for(int k=0;k<n-1;k++)

{

if(list[k]>list[k+1])

{

t=list[k];

list[k]=list[k+1];

list[k+1]=t;

}

}

}

}

void binarysearch(int key,int n,int list[])

{

int flag=1,low,mid,high;

low=0;

high=n-1;

while(low<=high)

{

mid=(low+high)/2;

if(key<list[mid])

high=mid-1;

else if (key>list[mid])

low=mid+1;

else if(key==list[mid])

{

cout<<"Search Successful!!\n";

cout<<"Element "<<key<<" is found at location "<<(mid+1)<<endl;

flag=0;

break;

}

}

if(flag)

cout<<"Search not Successful!!\n";

}

void output(int list[],int n)

{

cout<<"SORTED LIST IS: ";

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

{ cout<<list[i]<<" "; }

cout<<endl;

}

**OUTPUT:**

ENTER NUMBER OF ELEMENTS IN THE LIST: 6

ENTER THE ELEMENTS ONE BY ONE:

8 2 4 0 1 7

ENTER THE KEY TO BE SEARCHED: 0

SORTED LIST IS: 0 1 2 4 7 8

Search Successful!!

Element 0 is found at location 1