

Date Structures Lab Using C/C++

Subject Code: MCAL11

A Practical Journal Submitted in Fulfilment of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2023-2024

By

Mr. NeelKamal Bandgar

(Application Id: - 81113)

Under the Guidance of

Ms. Pooja



Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

University of Mumbai

PCP Center

[Shri Ram College of Commerce, Bhandup]



Institute of Distance and Open Learning
Vidyanagari, Kalina, Santacruz (E) -400098

CERTIFICATE

This to certify that, **NeelKamal Bandgar** appearing **Master in Computer Application (Semester I) Application ID: 81113** has satisfactorily completed the prescribed Practical of **MCAL11- Data Structures Lab Using C/C++** as laid down by the University of Mumbai for the academic year **2023- 24**

Teacher in charge

Examiners

Coordinator

IDOL, MCA

University of Mumbai

Date: - 29/01/24

Place: - Bhandup

Index

Module No	Practical	Page No.	Sign.
I	1.Implement program for Bubble Sort. 2.Implement program for Insertion Sort. 3.Implement program for Selection Sort.		
II	1.Implement program for Linear Search. 2.Implement program for Binary Search.		
III	Implement program for Stack using array.		
IV	Implementation program for Queue.		
V	Implementation program for Linked List.		
VI	Creating Binary Search Tree.		
VII	Graph creation using adjacency matrix.		

Module: 1

1] Bubble Sort:

Aim: Implement program for Bubble Sort.

Code:

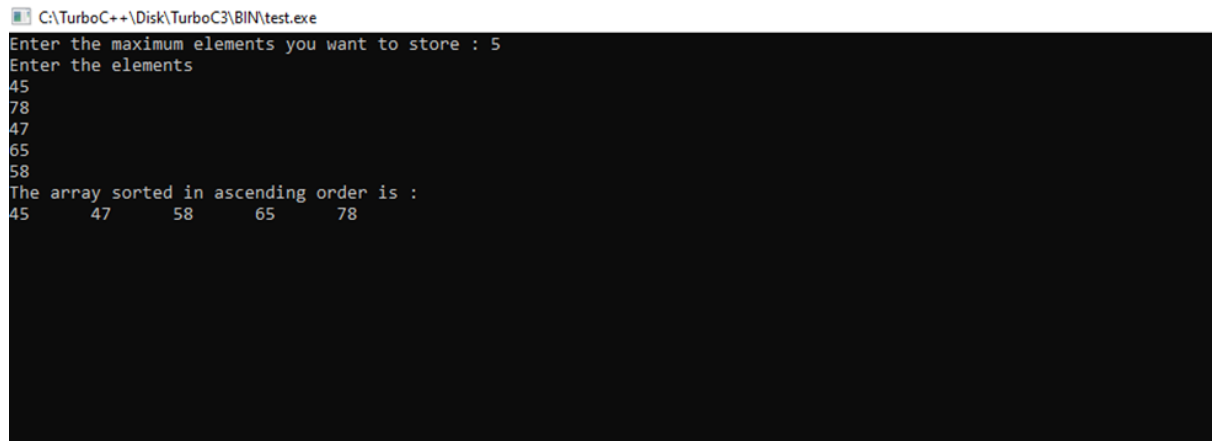
```
#include <stdio.h>
#include <conio.h>
int main()
{
    int i, n, temp, j, arr[10];
    printf("Enter the maximum elements you want to store : ");
    scanf("%d", &n);
    printf("Enter the elements \n");
    for(i=0;i<n;i++)
    {
        scanf("%d", &arr[i]);
    }

    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1;j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

    printf("The array sorted in ascending order is :\n");
    for(i=0;i<n;i++)
    printf("%d\t", arr[i]);
    getch();
}
```

```
return 0;
}
```

Output:

A screenshot of a TurboC++ IDE's output window. The title bar reads 'C:\TurboC++\Disk\TurboC3\BIN\test.exe'. The window contains the following text: 'Enter the maximum elements you want to store : 5', 'Enter the elements', followed by the input values '45', '78', '47', '65', and '58' on separate lines. Then it displays 'The array sorted in ascending order is :', followed by the sorted array '45', '47', '58', '65', and '78' on separate lines.

```
C:\TurboC++\Disk\TurboC3\BIN\test.exe
Enter the maximum elements you want to store : 5
Enter the elements
45
78
47
65
58
The array sorted in ascending order is :
45    47    58    65    78
```

2] Insertion Sort:

Aim: Implement program for Insertion Sort.

Code:

```
#include<stdio.h>

#include<conio.h>

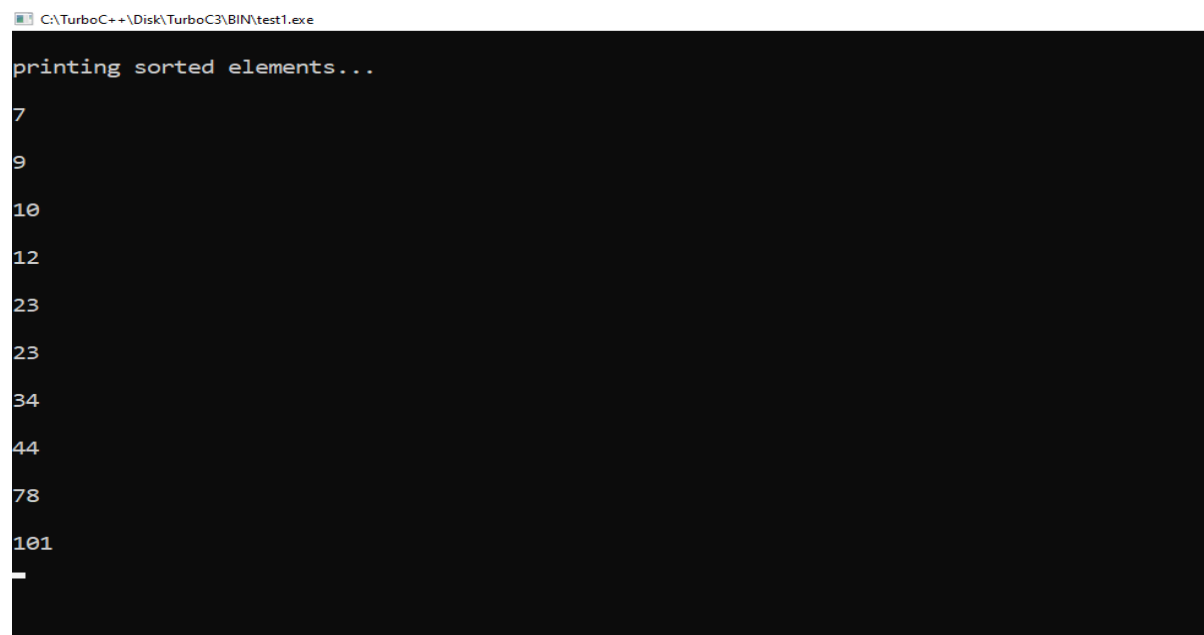
void main ()
{
int i, j, k,temp;
int a[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
printf("\nprinting sorted elements...\n");
for(k=1; k<10; k++)
{
temp = a[k];
j= k-1;
while(j>=0 && temp <= a[j])
{
a[j+1] = a[j];
j = j-1;
}
a[j+1] = temp;
```

```

}
for(i=0;i<10;i++)
{
printf("\n%d\n",a[i]);
}
getch();
}

```

Output:



```

C:\TurboC++\Disk\TurboC3\BIN\test1.exe
printing sorted elements...
7
9
10
12
23
23
34
44
78
101
_

```

3] Selection Sort:

Aim: Implement program for Selection Sort.

Code:

```

#include<stdio.h>

#include<conio.h>

int smallest(int[],int,int);

void main ()
{
int a[10] = {10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
int i,j,k,pos,temp;
for(i=0;i<10;i++)

```

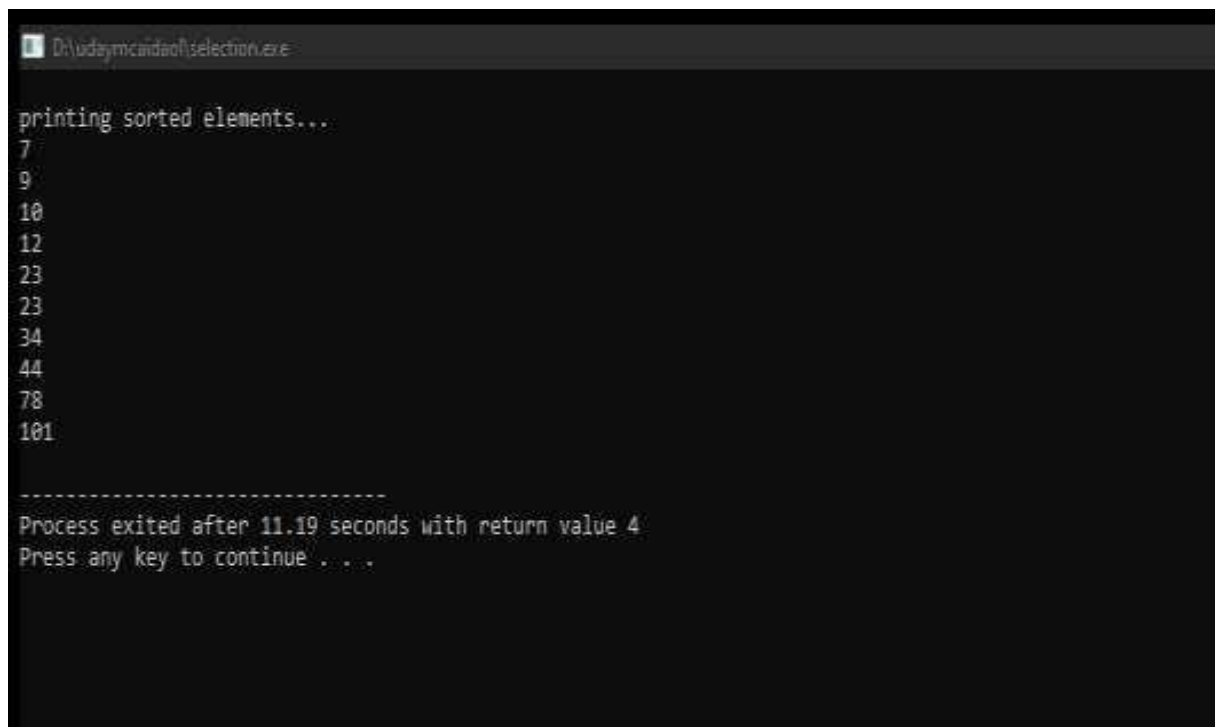
```

{
pos = smallest(a,10,i);
temp = a[i];
a[i]=a[pos];
a[pos] = temp;
}
printf("\nprinting sorted elements...\n");
for(i=0;i<10;i++)
{
printf("%d\n",a[i]);
}
}

int smallest(int a[], int n, int i)
{
int small,pos,j;
small = a[i];
pos = i;
for(j=i+1;j<10;j++)
{
if(a[j]<small)
{
small = a[j];
pos=j;
}
}
getch();
return pos;
}

```

Output:



A screenshot of a Windows command prompt window. The title bar at the top reads "D:\udaymca\dao\selection.exe". The command prompt shows the following text:

```
printing sorted elements...
7
9
10
12
23
23
34
44
78
101

-----
Process exited after 11.19 seconds with return value 4
Press any key to continue . . .
```


Module: - II

1) Linear Search:

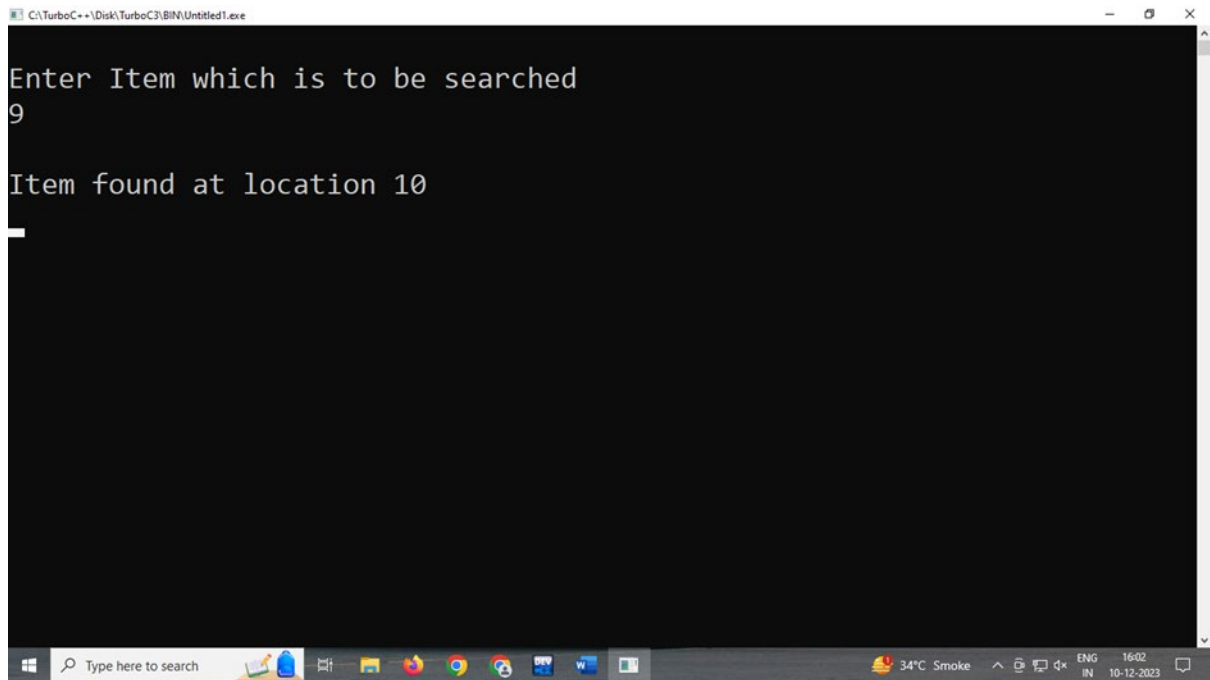
Aim: Implement program for Linear Search.

Code:

```
#include<stdio.h>
#include<conio.h>
void main ()
{
int a[10] = {10, 23, 40, 1, 2, 0, 14, 13, 50, 9};
int item, i, flag;
printf("\nEnter Item which is to be searched\n");
scanf("%d",&item);
for (i = 0; i < 10; i++)
{
if(a[i] == item)
{
flag = i+1;
break;
}
else
{
flag = 0;
}
}
if(flag != 0)
{
printf("\nItem found at location %d\n",flag);
}
else
```

```
{  
printf("\nItem not found\n");  
}  
getch();  
}
```

Output:



```
C:\TurboC++\BIN\Untitled1.exe  
Enter Item which is to be searched  
9  
Item found at location 10
```

2] Binary Search:

Aim: Implement program for Binary Search.

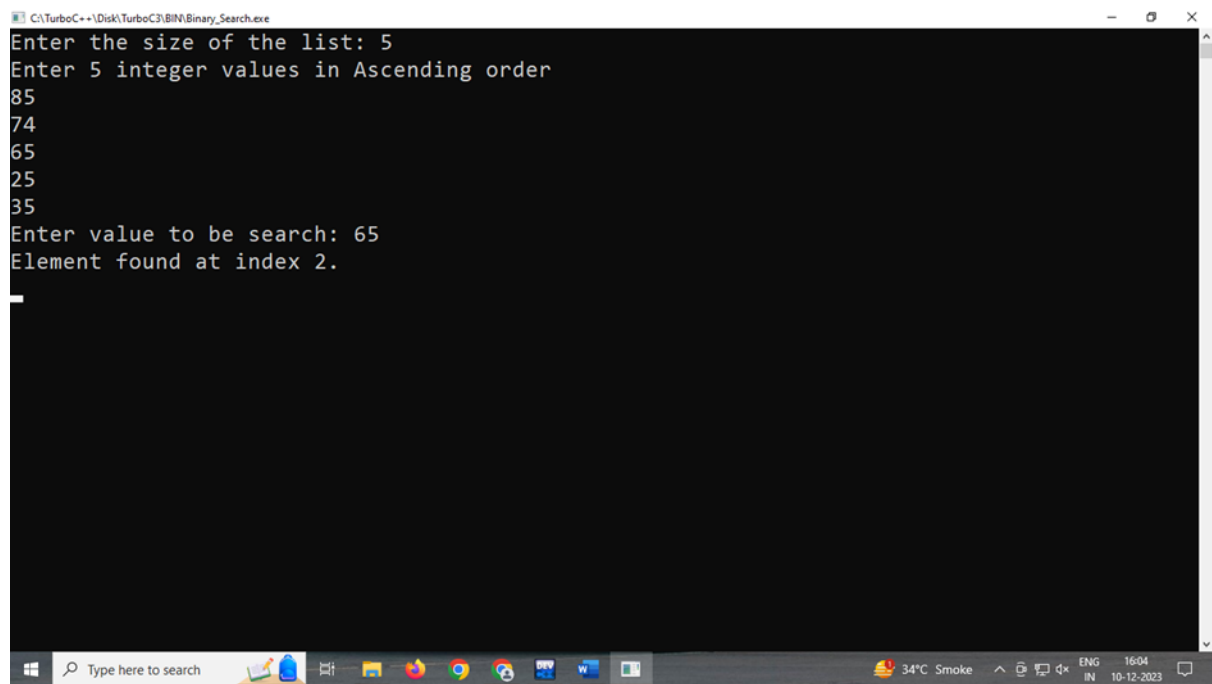
Code:

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
int first, last, middle, size, i, key, list[100];  
printf("Enter the size of the list: ");  
scanf("%d",& size);
```

```
printf("Enter %d integer values in Ascending order\n", size);
for (i = 0; i < size; i++)
{
scanf("%d",&list[i]);
}
printf("Enter value to be search: ");
scanf("%d", &key);
first = 0;
last = size - 1;
middle = (first+last)/2;
while (first <= last)
{
if (list[middle] <key)
{
first = middle + 1;
}
else if (list[middle] == key)
{
printf("Element found at index %d.\n",middle);
break;
}
else
{
last = middle - 1;
}
middle = (first + last)/2;
}
```

```
if (first > last)
{
printf("Element Not found in the list.");
}
getch();
}
```

Output:



The screenshot shows a TurboC++ console window titled "C:\TurboC++\1\Disk1\TurboC3\BIN\Binary_Search.exe". The program prompts the user to enter the size of the list (5) and then 5 integer values in ascending order (85, 74, 65, 25, 35). It then prompts for a value to be searched (65) and outputs "Element found at index 2." The Windows taskbar at the bottom shows the date and time as 16:04 on 10-12-2023.

```
C:\TurboC++\1\Disk1\TurboC3\BIN\Binary_Search.exe
Enter the size of the list: 5
Enter 5 integer values in Ascending order
85
74
65
25
35
Enter value to be search: 65
Element found at index 2.
```

Module: - III

1] Stack using array:

Aim: Implement program for Stack using array.

Code:

```
#include <stdio.h>

int stack[100],i,j,choice=0,n,top=-1;

void push();
void pop();
void show();

main ()
{
    printf("Enter the number of elements in the stack ");
    scanf("%d",&n);
    printf("***Stack operations using array***");
    printf("\n-----\n");
    while(choice != 4)
    {
        printf("Chose one from the below options...\n");
        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
        printf("\n Enter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
```

```

    }
    case 2:
    {
        pop();
        break;
    }
    case 3:
    {
        show();
        break;
    }
    case 4:
    {
        printf("Exiting....");
        break;
    }
    default:
    {
        printf("Please Enter valid choice ");
    }
    }
    }
}

void push ()
{
    int val;
    if (top == n )

```

```
printf("\n Overflow");
else
{
printf("Enter the value?");
scanf("%d",&val);
top = top +1;
stack[top] = val;
}
}
void pop ()
{
if(top == -1)
printf("Underflow");
else
top = top -1;
}
void show()
{
for (i=top;i>=0;i--)
{
printf("%d\n",stack[i]);
}
if(top == -1)
{
printf("Stack is empty");
}
}
```

Output:

 C:\TurboC++\Disk\TurboC3\BIN\Stack Using array.exe

Enter the number of elements in the stack 2

Stack operations using array

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

1

Enter the value?12

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

1

Enter the value?25

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

2

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Enter your choice

3

12

Chose one from the below options...

1.Push

2.Pop

3.Show

4.Exit

Module: - IV

1] Queue Array

Aim: Implementation of the queue array.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 50
int queue_array[MAX];
int rear = -1;
int front = -1;
insert()
{
int add_item;
if(rear==MAX-1)
printf("Queue Overflow\n");
else
{
if(front == -1)
front=0;
printf("Insert the element in queue:");
scanf("%d",&add_item);
rear = rear+1;
queue_array[rear]=add_item;
}
return 1;
}
```

```

deleteq()
{
if(front == -1 || front>rear)
{
printf("Queue Underflow\n");
return 1;
}
else
{
printf("Element deleted from queue is:\t");
printf("%d", queue_array[front]);
front=front+1;
}
return 1;
}

display()
{
int i;
if(front == -1 || front>rear)
{
printf("Queue is empty\n");
}
else
{
printf("Queue is : \n");
for(i=front;i<=rear;i++)
printf("%d",queue_array[i]);

```

```

printf("\n");
}
return 1;
}
main(){
int ch;
//clrscr();
while(1){
printf("\n1. Insert\n");
printf("2. Delete\n");
printf("3. Display\n");
printf("4. Exit\n");
printf("Enter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1: insert();
break;
case 2: deleteq();
break;
case 3: display();
break;
case 4: exit(0);
break;
default: printf("\n Wrong choice\n");
}
}
}

```

```
}
```

Output:

 C:\TurboC++\Disk\TurboC3\BIN\Ordinary Queue.exe

```
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Insert the element in queue:3
```

```
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Insert the element in queue:5
```

```
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Insert the element in queue:2
```

```
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:3
Queue is :
352
```

```
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:
```

Module: - V

1] Singly Linked List

Aim: Implementation of the Singly Linked List.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node* create(int n);
void display(struct node* head);

main()
{
    int n=0;
    // clrscr();
    struct node* head=NULL;
    printf("Enter the how many nodes");
    scanf("%d",&n);
    head=create(n);
    display(head);
    getch();
}

struct node* create(int n)
{

```

```

int i=0;
struct node* head=NULL;
struct node* temp=NULL;
struct node* p=NULL;
printf("\nEnter the element of the linklist");
for(i=0;i<n;i++)
{
temp=(struct node*)malloc(sizeof(struct node*));
scanf("%d",&temp->data);
temp->next=NULL;
if(head==NULL)
{
head=temp;
}
else
{
p=head;
while(p->next!=NULL)
p=p->next;
p->next=temp;
}
}
return head;
}
void display(struct node*head)
{
printf("linked list is ");

```

```
struct node* p=head;
while(p!=NULL)
{
printf(" %d",p->data);
printf("->");
p=p->next;
}
printf("NULL");
}
```

Output:

 C:\TurboC++\Disk\TurboC3\BIN\Single Linkedlist.exe

Enter the how many nodes

3

Enter the element of the linklist

25

14

85

linked list is 25-> 14-> 85->NULL

Process exited after 14.82 seconds with return value 0

Press any key to continue . . .

Module: - VI

1] Binary search tree

Aim: Creating Binary search tree.

Code:

```
#include<iostream>
#include<stdio.h>
#include<stdlib.h>

using namespace std;
void insert(int);
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *root;
int main ()
{
int choice,item;
do
{
cout<<"\nEnter the item which you want to insert?\n";
cin>>item;
insert(item);
cout<<"\n Press 0 to insert more? \n";
cin>>choice;
```



```

}while(choice == 0);
return 0;
}
void insert(int item)
{
struct node *ptr, *parentptr , *nodeptr;
ptr = (struct node *) malloc(sizeof (struct node));
if(ptr == NULL)
{
cout<<"cannot insert";
}
else
{
ptr -> data = item;
ptr -> left = NULL;
ptr -> right = NULL;
if(root == NULL)
{
root = ptr;
root -> left = NULL;
root -> right = NULL;
}
else
{
parentptr = NULL;
nodeptr = root;
while(nodeptr != NULL)

```

```

{
parentptr = nodeptr;
if(item < nodeptr->data)
{
nodeptr = nodeptr -> left;
}
else
{
nodeptr = nodeptr -> right;
}
}
if(item < parentptr -> data)
{
parentptr -> left = ptr;

}
else
{
parentptr -> right = ptr;
}
}
cout<<"Node Inserted";
}
}

```

Output:

C:\TurboC++\Disk\TurboC3\BIN\Binary Search Tree.exe

```
Enter the item which you want to insert?
12
Node Inserted
Press 0 to insert more?
0

Enter the item which you want to insert?
78
Node Inserted
Press 0 to insert more?
0

Enter the item which you want to insert?
98
Node Inserted
Press 0 to insert more?
```

Module: -VII

1] Adjacency matrix of graph:

Aim: Graph Creation using Adjacency matrix.

Code:

```
#include<iostream>

using namespace std;

int vertArr[20][20]; //the adjacency matrix initially 0

int count = 0;

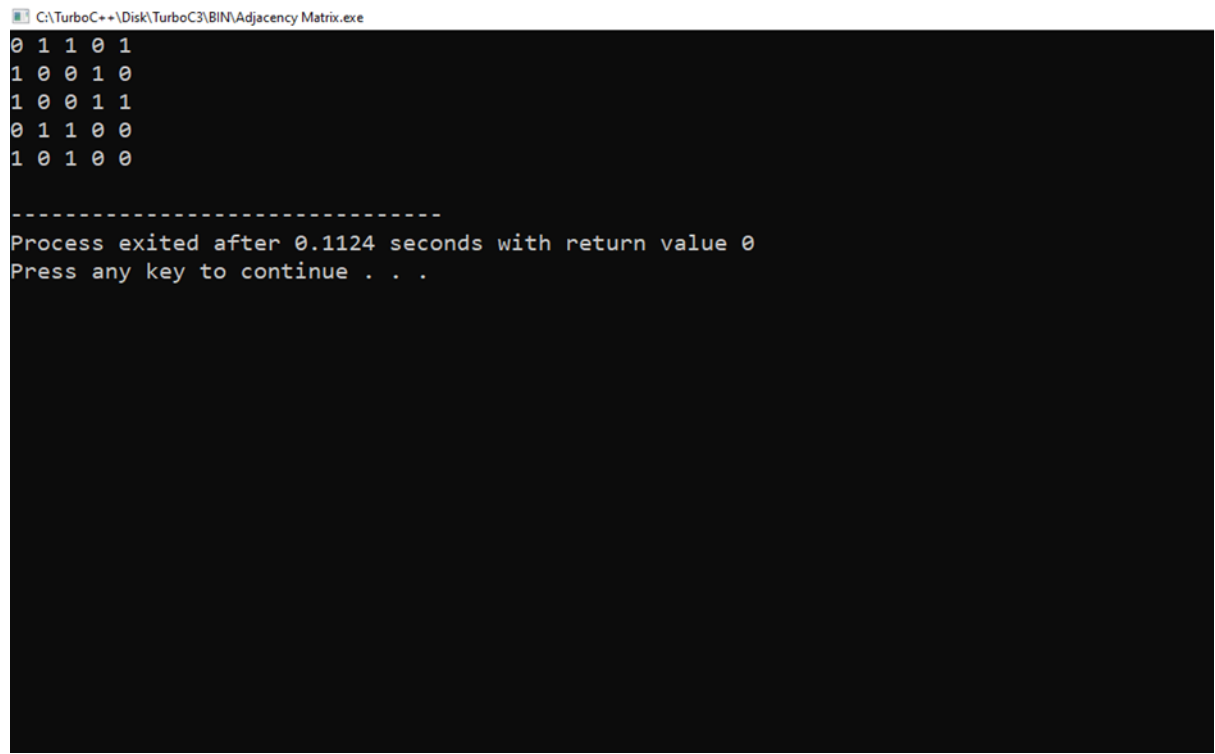
void displayMatrix(int v){
    int i,j;
    for(i=0;i<v;i++){
        for(j=0;j<v;j++){
            cout<<vertArr[i][j]<<" ";
        }
        cout<<endl;
    }
}

void add_edge(int u, int v){ //function to add edge into the matrix
    vertArr[u][v] =1;
    vertArr[v][u] =1;
}

int main(){
    int v = 5; // There are 6 vertices in the graph
    add_edge(0,1);
    add_edge(0,2);
    add_edge(0,4);
    add_edge(1,3);
```

```
add_edge(3,2);
add_edge(2,4);
displayMatrix(v);
return(0);
}
```

Output:



The screenshot shows a TurboC++ console window with the title bar "C:\TurboC++\Disk\TurboC3\BIN\Adjacency Matrix.exe". The output displays a 5x5 adjacency matrix as a grid of 0s and 1s. Below the matrix is a dashed line, followed by the text "Process exited after 0.1124 seconds with return value 0" and "Press any key to continue . . .".

```
C:\TurboC++\Disk\TurboC3\BIN\Adjacency Matrix.exe
0 1 1 0 1
1 0 0 1 0
1 0 0 1 1
0 1 1 0 0
1 0 1 0 0

-----
Process exited after 0.1124 seconds with return value 0
Press any key to continue . . .
```