**/\* Program 1**

**Write a program to accept the elements in 2D array and perform all the matrix operations i.e. addition, multiplication, transpose etc.\*/**

#include<iostream.h> #include<conio.h> #define mr 10

#define mc 10 main()

{

int i,j,k;

int m1[mr][mc],m2[mr][mc],m3[mr][mc],m4[mr][mc],m5[mr][mc]; int r1,r2,c1,c2;

clrscr();

cout<<"Enter no.of rows for matrix 1 :"<<endl; cin>>r1;

cout<<"Enter no of coloumns for matrix 1 :"<<endl; cin>>c1;

cout<<"Enter data for matrix 1 :"<<endl; for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++)

{

cout<<"Enter element :"; cin>>m1[i][j];

}

}

cout<<"Enter no.of rows for matrix 2 :"<<endl; cin>>r2;

cout<<"Enter no of coloumns for matrix 2 :"<<endl; cin>>c2;

cout<<"Enter data for matrix 2 :"<<endl; for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++)

{

cout<<"Enter element :"; cin>>m2[i][j];

}

}

/\*addition \*/ for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++) m3[i][j]=m1[i][j]+m2[i][j];

}

/\*transpose \*/

for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++) m4[j][i]=m1[i][j];

}

/\*Multiplication \*/

if(c1!=r2)

{

cout<<"Multiplication not possible"; getch();

}

else

{

// initialise m5 to 0 for(i = 0;i<r1;i++)

for(j=0;j<c2;j++) m5[i][j]=0;

for(k=0;k<r1;k++)

{

for(i=0;i<c2;i++)

{

for(j=0;j<c1;j++) m5[k][i]=m5[k][i]+m1[k][j]\*m2[j][i];

}

}

}

cout<<"Matrix 1 is :"<<endl; for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++) cout<<"\t"<<m1[i][j]; cout<<endl;

}

cout<<"Matrix 2 is :"<<endl; for(i=0;i<r2;i++)

{

for(j=0;j<c2;j++) cout<<"\t"<<m2[i][j]; cout<<endl;

}

cout<<"Addition of matrix is :"<<endl; for(i=0;i<r1;i++)

{

for(j=0;j<c1;j++) cout<<"\t"<<m3[i][j]; cout<<endl;

}

cout<<"Transpose of matrix is :"<<endl; for(i=0;i<c1;i++)

{

for(j=0;j<r1;j++) cout<<"\t"<<m4[i][j]; cout<<endl;

}

cout<<"Multiplication of matrix is :"<<endl; for(i=0;i<r1;i++)

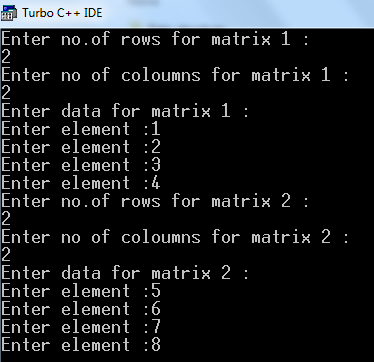
{

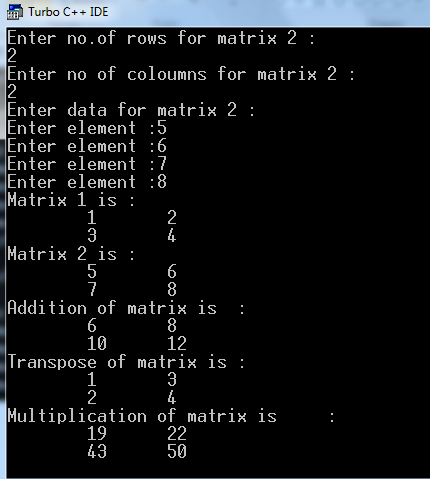
for(j=0;j<c2;j++) cout<<"\t"<<m5[i][j]; cout<<endl;

}

getch();

}





**/\* Program 2**

**Explain following techniques**

# Bubble Sort

This is the simplest kind of sorting method. We do this bubble sort procedure in several iterations, which are called passes.

Algorithm

1. Read the total number of elements say n.
2. Store the elements in the array.
3. Set the i=0
4. Compare the adjacent elements
5. Repeat step 4 for all n elements
6. Increment the value of I by 1 and repeat step 4,5 for i<n.
7. Print the sorted list of elements.
8. Stop

Complexity

Time complexity: O (n2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| e.g. | 10 | 30 | 20 | 50 | 40 |
| Pass 1: | 10 | 30 | 20 | 50 | 40 |
| 10 | 30 | 20 | 50 | 40 |  |
| 10 | 20 | 30 | 50 | 40 |  |
| 10 | 20 | 30 | 50 | 40 |  |
| Pass 2: | 10 | 20 | 30 | 50 | 40 |
|  | 10 | 20 | 30 | 50 | 40 |
|  | 10 | 20 | 30 | 50 | 40 |
| Pass 3: | 10 | 20 | 30 | 50 | 40 |
|  | 10 | 20 | 30 | 50 | 40 |
| Pass 4: | 10 | 20 | 30 | 50 | 40 |
| Sorted Array: | 10 | 20 | 30 | 40 | 50 |

# Insertion sort

In this, the elements are inserted at their appropriate place. Hence is the name insertion sort.

Algorithm:

For i 1 to n-1 do

{

Temp A[i]

j i-1

while(j>=0)AND (A[j]>temp)do

{

A[j+1] A[i]

j j-1

}

A[j+1] Temp

}

e.g. 77 33 44 11 88 22 66 55

Pass 1: 44 11 88 22 66 55

33

77

Pass 2: 11 88 22 66 55

33

44

77

Pass 3: 88 22 66 55

11

33

44

77

Pass 4: 22 66 55

11

33

44

77

88

Pass 5: 66 55

11

22

33

44

77

88

Pass 6: 55

11

22

33

44

66

77

88

Pass 7: 11 22 33 44 55 66 77 88

# Radix sort

In this sorting records or elements of the table are arranged in sequential order.

Complexity:

Best case: O (n log n) Worst case: O (n2)

348, 143, 361, 423, 538, 128, 321, 543, 366

Pass 1:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | 543 |  |  |  |  | 128 |  |
|  | 321 |  | 423 |  |  |  |  | 538 |  |
|  | 361 |  | 143 |  |  | 366 |  | 348 |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Elements: 361, 321, 143, 423, 543, 366, 348, 538, 128

Pass 2:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 128 |  | 348 |  |  |  |  |  |
|  |  | 423 |  | 543 |  | 366 |  |  |  |
|  |  | 321 | 538 | 143 |  | 361 |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Elements: 321, 423, 128, 538, 143, 543, 348, 361, 366

Pass 3:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 366 |  |  |  |  |  |  |
|  |  |  | 361 |  |  |  |  |  |  |
|  | 143 |  | 348 |  | 543 |  |  |  |  |
|  | 128 |  | 321 | 423 | 538 |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Sorted elements: 128, 143, 321, 348, 361, 366, 423, 538, 543

**/\*Program 3**

**Suppose an array contains n elements. Given a number x that may occur several times in the array. Write a program to find**

* 1. **The number of occurrences of x in the array.**
  2. **The position of first occurrence of x in the array.\*/**

#include<iostream.h> #include<conio.h> #define mx 10

main()

{

int i,m[mx], noe, kd, flag=0,count=0,firstl=-1; clrscr();

cout<<"Enter no of elements :"; cin>>noe;

cout<<"Enter data :"; for(i=0;i<noe;i++)

{

cout<<"Enter element :"; cin>>m[i];

}

cout<<"Enter the element whose occurrences are to be counted :"; cin>>kd;

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

{

if(kd==m[i])

{

flag=1; count++; if(firstl==-1)

firstl=i;

}

}

if(flag==0)

cout<<kd<<”does not exists”<<endl; else

{

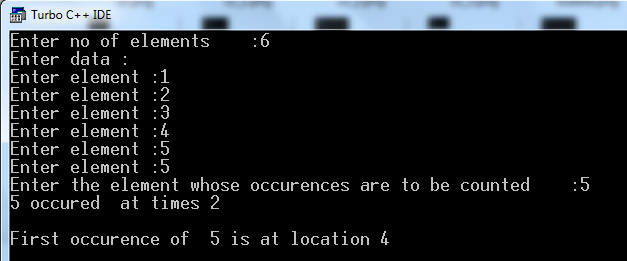
cout<<kd<<" occurred "<<count<<”times :"<<endl;

cout<<"First occurrence of "<<kd<<" is at location :"<<firstl<<endl;

}

getch();

}



**/\*Program 4 :**

**Write a program in CPP to delete particular element from an array of 10 integers.\*/**

#include<iostream.h> #include<conio.h> void main()

{

clrscr();

int arr[50], size, i, del, count=0; cout<<"Enter array size : "; cin>>size;

cout<<"Enter array elements : "; for(i=0; i<=size; i++)

{

cin>>arr[i];

}

cout<<"Enter element to be delete : "; cin>>del;

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

{

if(arr[i]==del)

{

for(int j=i; j<(size-1); j++)

{

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

}

count++; break;

}

}

cout<<"original array is :"<<endl; for(i=0; i<=size; i++)

{

cout<<" \t"<<arr[i]<<endl;

}

if(count==1)

{

}

else

{

cout<<"Element not found..!!";

cout<<"\nElement deleted successfully..!!\n"; cout<<"Now the new array is :\n";

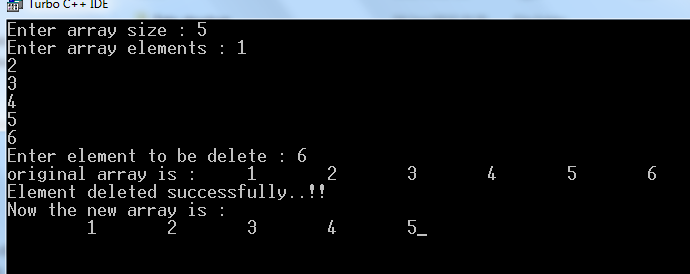
for(i=0; i<size; i++)

{

}

}getch();}

cout<<arr[i]<<" ";



**/\* Program 5**

**Consider two single dimensional array of size 20 and 3 respectively. Write a program in CPP to display all the elements which are common in both arrays. \*/**

#include<iostream.h> #include<conio.h> #define mx 10

main()

{

int i,j,n1,n2,m1[mx],m2[mx]; clrscr();

cout<<"Enter no.of elements for array 1 :"<<endl; cin>>n1;

cout<<"Enter data :"<<endl; for(i=0;i<n1;i++)

{

cout<<"Enter element :"<<endl; cin>>m1[i];

}

cout<<"Enter no.of elements for array 2 :"<<endl; cin>>n2;

cout<<"Enter data"; for(i=0;i<n2;i++)

{

cout<<"Enter element :"; cin>>m2[i];

}

for(i=0;i<n1;i++)

{

for(j=0;j<n2;j++)

{

if(m1[i]==m2[j])

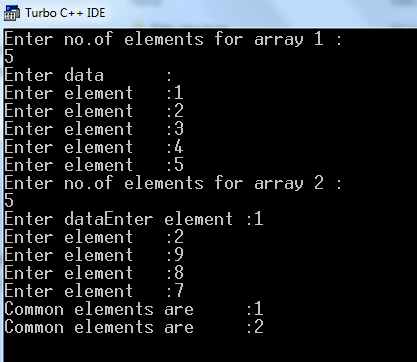
cout<<"Common elements are :"<<m1[i]<<endl;

}

}

getch();

}



**/\*Program 7**

**Write a menu driven program for stack contain following function**

* **PUSH**
* **POP**
* **DISPLAY**
* **PEEK**

**\*/**

#include<iostream.h> #include<conio.h> #include<stdlib.h> #include<ctype.h> #define mx 10

int stk[mx];

int top=-1, stackempty=1,stackfull=0;

void push(int); int pop(void);

int printmenu(void);

main()

{

int puc,poc,resp,resp1,ch; do

{

clrscr(); ch=printmenu(); switch(ch)

{

case 1: do

{

cout<<"\nEnter the number :"; cin>>puc;

push(puc);

cout<<"\nAny more additions(y/n)"; resp1=getche();

}while(toupper(resp1)=='Y'); break;

case 2:

if(stackempty==-1) cout<<”Stack is empty”;

else

{

cout<<"\nThe stack is \n"; while(stackempty!=1)

{

poc=pop(); cout<<"\t"<<poc;

}

}

break;

case 3:

cout<<"The peeked element is \n";

if(stackempty==1) cout<<"Stack is empty!";

else

{

}

poc=stk[top];

cout<<"Peeked char is"<<poc;

break;

case 4:

exit(1);

}

getch();

cout<<"Any more trials(y/n)"; resp=getche();

}while(toupper(resp)=='Y');

}

void push(int puc)

{

if(top==mx-1)

{

else

{

}

cout<<"Stack is full"; getch();

exit(1);

}

++top; stk[top]=puc; stackempty=0; if(top==mx-1)

stackfull=1;

}

int pop(void)

{

int poc; if(stackempty==1)

{

}

else

{

}

}

cout<<"Stack is empty"; getch();

poc=stk[top]; top--; if(top==-1)

stackempty=1; return(poc);

int printmenu(void)

{

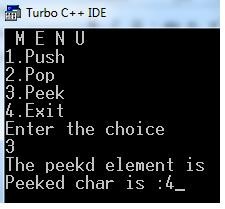
int ch;

cout<<" M E N U"<<endl; cout<<"1. Push "<<endl; cout<<"2.Pop "<<endl; cout<<"3.Peek "<<endl; cout<<"4.Exit "<<endl; cout<<"Enter the choice"<<endl; cin>>ch;

return(ch);

}





**/\*Program 9**

**Write a program in CPP to implements Queue using Array. \*/**

#include<iostream.h> #include<conio.h> #include<ctype.h> #include<stdlib.h> #define mx 20

int q[mx];

int r = -1, f=-1;

/\* prototypes \*/ int qfull(void);

int qempty(void); void addq(void); void delq(void); void displayq(void);

main()

{

char resp; int ch;

do

{

clrscr();

cout<<" MENU\n"; cout<<"1. ADD\n"; cout<<"2, Delete\n"; cout<<"3. Display\n"; cout<<"4. Exit\n"; cout<<"Enter the choice\n"; cin>>ch;

switch(ch)

{

case 1 : addq(); break;

case 2 : delq(); break;

case 3 : displayq(); break; case 4:

exit(1);

default :

cout<<"Invalid Choice... 1..4";

}

cout<<"Any more trials(y/n) "; resp=getche();

}while(toupper(resp)=='Y');

}

int qfull(void)

{

if(r==mx-1) return(1);

else return(0);

}

int qempty(void)

{

if(r==-1 && f==-1) return(1);

else

return(0);

}

void addq(void)

{

char resp; do

{

if(qfull()==1)

{

cout<<"Queue is full"; getch();

}

else

{

cout<<"\nEnter the element :"; cin>>q[++r];

/\* sp case\*/

if(f==-1) f=0;

}

cout<<"\n Do you want to add any more elements(y/n)"; resp=getche();

}while(toupper(resp)=='Y');

}

void delq(void)

{

char resp; do

{

if(qempty()==1)

{

cout<<"Queue is empty"; getch();

}

else

{

cout<<"deleted element is : "<<q[f++]; getch();

}

cout<<"Any more deletions(y/n)"; resp=getche();

}while(toupper(resp)=='Y');

}

void displayq(void)

{ int i;

if(qempty()==1) cout<<"Queue is empty";

else

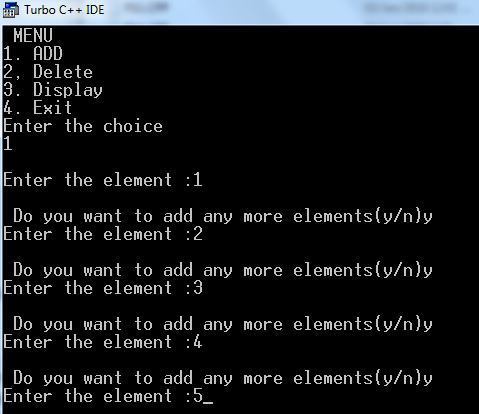
{

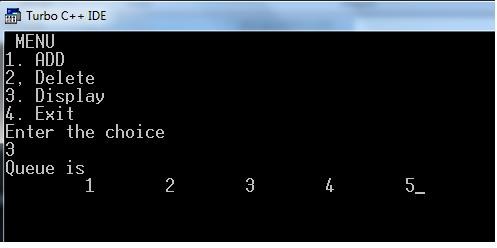
cout<<"Queue is \n"; for(i = f;i<=r;i++) cout<<"\t"<<q[i];

}

getch();

}





**/\*Program 10**

**Consider the single Linked List contains following elements: Roll no int, sname char (20), city char (20) and course char (3).**

**Write a program, in CPP to represent linked List with the above elements.**

**\*/**

#include<iostream.h> #include<conio.h> #include<alloc.h> #include<stdlib.h> #include<ctype.h> struct stud

{

int rno;

char sname[20]; char city[20]; char course[3]; struct stud \*link;

}\*h,\*p,\*n,\*c;

main()

{

char resp; int s;

h=p=c=n=NULL; clrscr();

do

{

//c= (stud \*) malloc(sizeof(struct stud));

//s=sizeof(struct stud);

c=new stud;

cout<<"\nEnter the Rollno :"; cin>>c->rno;

cout<<"Enter the name :"; cin>>c->sname; cout<<"Enter the city :"; cin>>c->city;

cout<<"Enter the Courese :"; cin>>c->course;

c->link= NULL; if(h==NULL)

{ h=c; p=c;

}

else

{

p->link=c; p=c;

}

cout<<"Any more additions(y/n):"; resp=getche();

}while(toupper(resp)=='Y');

cout<<"\nLinked List is "<<endl; n=h;

while(n!=NULL)

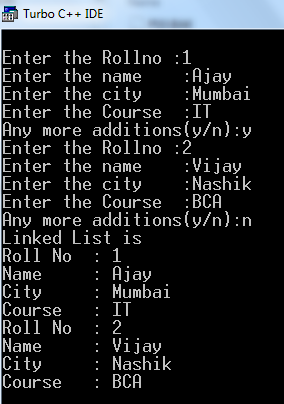
{

}

getch();

}

cout<<"Roll No : "<<n->rno<<endl; cout<<"Name : "<<n->sname<<endl; cout<<"City : "<<n->city<<endl; cout<<"Course : "<<n->course<<endl; n=n->link;



**/\*program 11**

**Write menu driven program which Create and display the circular linked list.**

**\*/** #include<iostream.h> #include<conio.h> #include<malloc.h> #include<stdlib.h> #include<ctype.h>

struct node

{

int data;

struct node\*link;

}\*r,\*f;

int printmenu(void); void addq(void); void delq(void); void displayq(void);

main()

{

int ch; char resp; clrscr();

r=f= NULL;

do

{

ch = printmenu(); switch(ch)

{

case 1:

addq(); break;

case 2:

delq(); break;

case 3:

displayq(); break; case 4:

exit(1);

default :

cout<<"Invalid choice... Range 1.. 4";

}

cout<<"\n Any more trials(y/n) : "; resp=getche();

}while(toupper(resp)=='Y');

}

int printmenu(void)

{

int ch; cout<<"MENU"<<endl; cout<<"1.Add"<<endl; cout<<"2.Delete"<<endl; cout<<"3.Display"<<endl; cout<<"4.Exit"<<endl; cout<<"\nEnter the choice"; cin>>ch;

return(ch);

}

void addq(void)

{

int num; struct node \*t; char resp;

do

{

cout<<"\nEnter the element :"; cin>>num;

//t=(node \*) malloc(sizeof(struct node)); t=new node;

t->data=num;

// if queue is empty if(r==NULL)

{

t->link= t; r=t;

}

else

{

t->link = r->link; r->link= t;

r=t;

}

cout<<"\n Any more additions(y/n):"; resp=getche();

}while(toupper(resp)=='Y');

}

void delq(void)

{

struct node \*t; if(r==NULL)

{

cout<<"Queue is empty!!"; getch();

}

else

{

// if one node t= r->link; if(t==r)

{

cout<<"\n deleted element is :\n"<<t->data; free(t);

r=NULL;

}

else

{

cout<<"\n deleted element is : "<<t->data; r->link = t->link;

free(t);

}

}

}

void displayq(void)

{

struct node \*t;

if(r==NULL)

{

cout<<"Queue is empty!!"; getch();

}

else

{

cout<<"\nQueue is \n"; t=r->link;

while(t!=r)

{

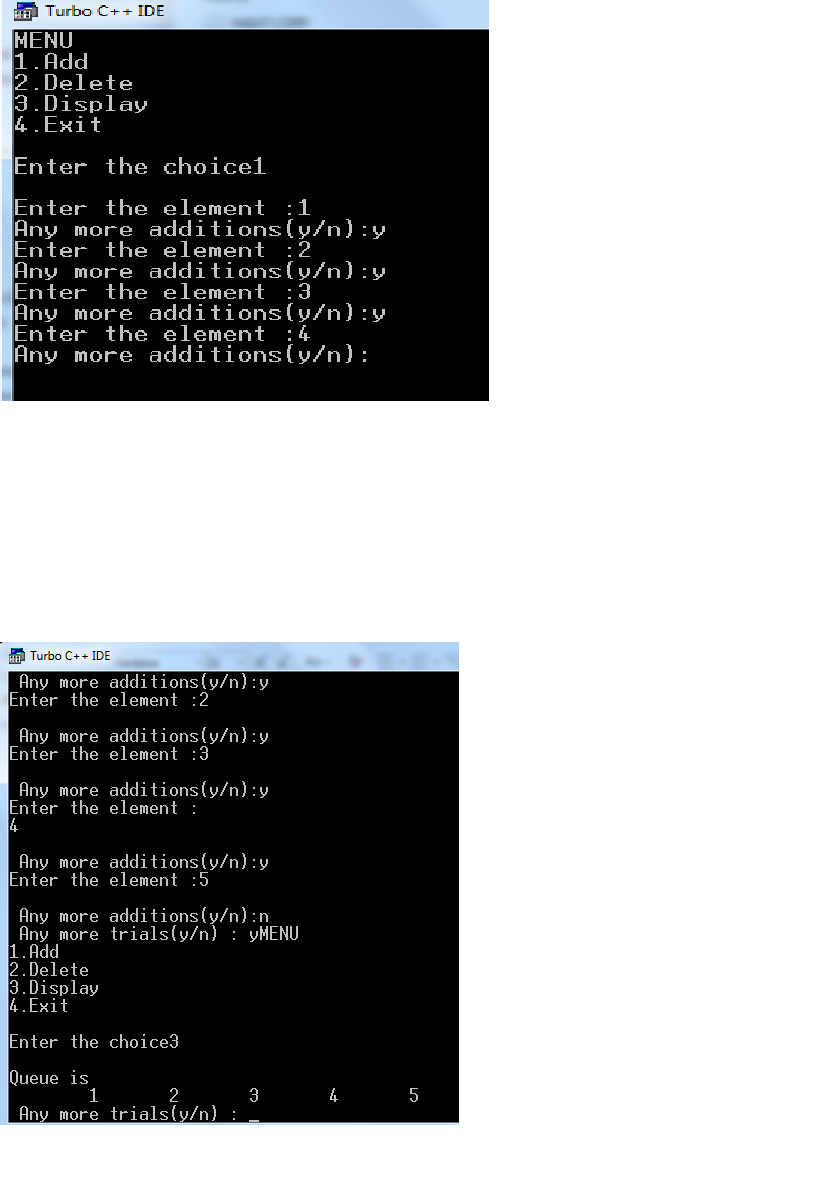
cout<<""<<t->data; t=t->link;

}

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

}

}



**/\*Program: 14**

**Write a program to traverse the graph using BFS method**

**\*/** #include<iostream.h> #include<conio.h> #include<alloc.h> #include<ctype.h> struct graph

{

int node;

struct graph \*down; struct graph \*next;

};

struct graph \*start,\*p,\*q,\*r,\*s; int que[10],visit[10],front,rear; void addnode(int x)

{

rear++; que[rear]=x; if(visit[x]==0) visit[x]=1;

}

int del()

{

int y; if(((rear==9)&&(front==rear))||(front==rear+1))

{

y=que[front]; front=0; rear=-1;

}

else

{

y=que[front]; front ++;

}

return (y);

}

void print()

{

p=q=start;

cout<<"Your graph is\n"; while(p!=NULL)

{

q=p->next; cout<<""<<p->node; while(q!=NULL)

{

q=q->next; cout<<""<<q->node; q=q->next;

}

p=p->down;

}

}

void bfs()

{

int inode,parent,flag; r=q=p=s=start;

cout<<"Enter the initial code\n"; cin>>inode;

cout<<""<<inode; addnode(inode);

while((rear!=-1)||(front!=rear+1))

{

parent=del(); p=start; flag=0; while(p!=NULL)

{

if(p->node==parent)

{

q=p->next; while(q!=NULL)

{

if(visit[q->node]==0)

{

cout<<""<<q->node; addnode(q->node); flag=1;

}

q=q->next;

}

}

if(flag==1) break;

else

p=p->down;

}

}

cout<<"";

}

struct graph \*create()

{

struct graph \*f;

f=(struct graph \*)malloc(sizeof(struct graph)); f->next=NULL;

f->down=NULL; cin>>f->node; return(f);

}

void main()

{

int no; char ch; clrscr();

front=0,rear=-1;

cout<<"Enter the graph structure for BFS\n"; do

{

cout<<"Enter a parent node\n"; p=create();

if(start==NULL)

{

start=p; r=q=p;

}

else

{

r->down=p; r=p;

}

s=p;

cout<<"Enter the adjacent node of\n"<<p->node; do

{

q=create(); s->next=q; s=q;

cout<<"Continue Adjacent node of(Y/N)\n"<<p->node; ch=getche();

}while(ch=='y');

cout<<"Continue entering the paret node(Y/N)\n"; ch=getche();

}while(ch=='Y'); print();

bfs();

}

**/\*Program 15:**

**Write a program to traverse the graph using DFS method.**

**\*/**

#include<iostream.h> #include<conio.h> #include<alloc.h> #include<ctype.h> struct graph

{

int node;

struct graph \*down; struct graph \*next;

};

struct graph \*start,\*p,\*q,\*r,\*s;

int stack[10],visit[10],tos=0,count;

void push(int x)

{

stack[tos]=x; tos++; if(visit[x]==0)

{

visit[x]=1; count++;

}

}

int pop()

{

int y; tos--;

y=stack[tos]; return (y);

}

void print()

{

p=q=start;

cout<<"Your graph is\n"; while(p!=NULL)

{

q=p->next; cout<<""<<p->node;

while(q!=NULL)

{

cout<<"%d"<<q->node; q=q->next;

}

p=p->down;

}

}

void dfs()

{

int is,parent,flag;

r=p=q=s=start;

cout<<"Enter the initial code"; cin>>is;

cout<<""<<is; push(is); while(tos!=1)

{

flag=0; parent=pop(); p=start; while(p!=NULL)

{

if(p->node==parent)

{

q=p->next; while(q!=NULL)

{

if(visit[q->node]==0)

{

cout<<""<<q->node; push(p->node); push(q->node); flag=1;

break;

}

else

q=q->next;

}

}

if(flag==1) break;

else

p=p->down;

}

}

cout<<")";

}

struct graph \*create()

{

struct graph \*f;

f=(struct graph \*)malloc(sizeof(struct graph)); f->next=NULL;

f->down=NULL; cin>>f->node; return(f);

}

void main()

{

int no; char ch; clrscr(); count=0;

cout<<"Enter the graph structure for DFS\n"; do

{

cout<<"Enter a parent node\n"; p=create();

if(start==NULL)

{

start=p; r=q=p;

}

else

{

r->down=p; r=p;

}

s=p;

cout<<"Enter the adjacent node of\n"<<p->node; do

{

q=create(); s->next=q; s=q;

cout<<"Continue Adjacent node of(Y/N)"; ch=getche();

}while(ch=='y');

cout<<"Continue entering the paret node(Y/N)"; ch=getche();

}while(ch=='Y');

}