**C++ PROGRAMMING**

**PROGRAM NO: 1**

**AIM: Program to calculate sum of two numbers using class.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class add\_num

{

int a,b;

public:

void read()

{

cout<<"Enter two numbers:";

cin>>a>>b;

}

void add()

{

cout<<"Sum =";

int sum;

sum=a+b;

cout<<sum<<"\n";

}

}ob;

main()

{

ob.read();

ob.add();

}

**OUTPUT:**

Enter two numbers:6 81

Sum =87

**PROGRAM NO: 2**

**AIM: Program to add points using class.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class point

{

int x,y;

public:

void read();

void display();

point add(point);

}p1,p2,p3;

void point ::read()

{

cin>>x>>y;

}

void point::display()

{

cout<<x<<" "<<y<<"\n";

}

point point:: add(point p1)

{

p3.x=x+p1.x;

p3.y=y+p1.y;

return p3;

}

main()

{

cout<<"Enter first point\n";

p1.read();

cout<<"Enter second point\n";

p2.read();

cout<<"First and second point is\n";

p1.display();

p2.display();

cout<<"sum is\n";

p3=p2.add(p1);

p3.display();

}

**OUTPUT:**

Enter first point

1 3

Enter second point

2 3

First and second point is

1 3

2 3

sum is

3 6

**PROGRAM NO: 3**

**AIM: Write a program to copying a string to another and concatenating two strings.**

**PROGRAM:**

#include<iostream>

#include<cstring>

usingnamespace std;

class strng

{

char str1[20];

char str2[20];

public:

strng(char \*s1,char \*s2)

{

strcpy(str1,s1);

strcpy(str2,s2);

}

void copy()

{

strcpy(str1,str2);

cout<<"Copied string str1=";

cout<<str1;

cout<<"\n";

}

void concat()

{

strcat(str1,str2);

cout<<"concatenated string str1=";

cout<<str1;

cout<<"\n";

}

void display()

{

cout<<str1<<"\n"<<str2<<"\n";

}

};

main()

{

cout<<"Two strings are:\n";

strng s("hai","hello");

s.display();

s.copy();

s.concat();

}

**OUTPUT:**

Two strings are:

hai

hello

Copied string str1=hello

concatenated string str1=hellohello

**PROGRAM NO: 4**

**AIM: Write a program to create to add two distances.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class distance

{

int feet;

float inch;

public:

distance()

{

feet=0;

inch=0;

}

void read();

void display();

void add(distance,distance);

}d1,d2,d3;

void distance::read()

{

cout<<"Enter feet and inch :";

cin>>feet>>inch;

}

void distance::display()

{

cout<<"feet="<<feet<<" ";

cout<<"inch="<<inch<<" ";

}

void distance::add(distance a,distance b)

{

feet=a.feet+b.feet;

inch=a.inch+b.inch;

if(inch>12)

{

feet++;

inch=inch-12;

}

}

main()

{

d1.read();

d2.read();

d1.display();

cout<<"\n";

d2.display();

d3.add(d1,d2);

cout<<"\nSum of feet and inch is:\n";

d3.display();

cout<<"\n";

}

**OUTPUT:**

Enter feet and inch :4 9

Enter feet and inch :10 5

feet=4 inch=9

feet=10 inch=5

Sum of feet and inch is:

feet=15 inch=2

**PROGRAM NO: 5**

**AIM: Write a program to create a stack and perform stack operations.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

#define size 100

class stack

{

int stck[size];

int tos;

public:

stack()

{

tos=0;

cout<<"Stack initialized\n";

}

~stack()

{

cout<<"\nStack destroyed\n";

}

void push(int i)

{

if(tos==size)

{

cout<<"stack fill\n";

}

stck[tos]=i;

tos++;

}

int pop()

{

if(tos==0)

{

cout<<"Stack underflow\n";

}

tos--;

return stck[tos];

}

};

main()

{

stack a,b;

a.push(1);

b.push(2);

a.push(3);

b.push(4);

cout<<a.pop()<<" ";

cout<<b.pop()<<" ";

cout<<a.pop()<<" ";

cout<<b.pop()<<" ";

}

**OUTPUT:**

Stack initialized

Stack initialized

3 4 1 2

Stack destroyed

Stack destroyed

**PROGRAM NO: 6**

**AIM: Write a C++ program to find the largest of three numbers using inline function.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

inlinevoid large(int a,int b,int c)

{

if(a>b)

{

if(a>c)

{

cout<<a<<" is largest\n";

}

else

{

cout<<c<<" is largest\n";

}

}

else

{

if(b>c)

{

cout<<b<<" is largest\n";

}

else

{

cout<<c<<" is largest\n";

}

}

}

main()

{

int a,b,c;

cout<<"Enter three numbers\n";

cin>>a>>b>>c;

large(a,b,c);

}

**OUTPUT:**

Enter three numbers

45 6 89

89 is largest

**PROGRAM NO: 7**

**AIM: Write a program to add two complex number using friend function.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class complex

{

int a,b;

public:

void read()

{

cout<<"Enter comlex number:";

cin>>a>>b;

}

void display()

{

cout<<a<<"+"<<b<<"i"<<"\n";

}

friend complex sum(complex,complex);

};

complex sum(complex ob1,complex ob2)

{

complex temp;

temp.a=ob1.a+ob2.a;

temp.b=ob1.b+ob2.b;

return temp;

}

main()

{

complex ob1,ob2,ob3;

ob1.read();

ob2.read();

cout<<"Two complex numbers are:\n";

ob1.display();

ob2.display();

cout<<"Sum=";

ob3=sum(ob1,ob2);

ob3.display();

}

**OUTPUT:**

Enter comlex number:3 4

Enter comlex number:5 3

Two complex numbers are:

3+4i

5+3i

Sum=8+7i

**PROGRAM NO: 8**

**AIM: Write a program to find minimum of two values for demonstrate friend classes**.

**PROGRAM:**

#include<iostream>

usingnamespace std;

class twovalues

{

int a;

int b;

public:

twovalues(int i,int j)

{

a=i;

b=j;

}

friendclass Min;

};

class Min

{

public:

int min(twovalues x);

};

int Min::min(twovalues x)

{

return x.a<x.b?x.a:x.b;

}

main()

{

twovalues ob(10,29);

Min m;

cout<<"Minimum vale is:";

cout<<m.min(ob)<<"\n";

}

**OUTPUT:**

Minimum vale is:10

**PROGRAM NO: 9**

**AIM: Write a program to find volume of cube , cylinder and rectangle using function overloading.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class fun

{

public:

int volume(int);

float volume(float,float);

float volume(float,float, float);

};

int fun::volume(int a)

{

return(a\*a\*a);

}

float fun::volume(float r,float h)

{

return(3.14\*r\*r\*h);

}

float fun::volume(float l,float b,float h2)

{

return(l\*b\*h2);

}

main()

{

int a,l,b,h,r,h2;

fun ob;

cout<<"Ente side of a cube:";

cin>>a;

cout<<"Volume of a cube is="<<ob.volume(a)<<"\n";

cout<<"Enter radious and height of cylinder:";

cin>>r>>h;

cout<<"Volume of cylinder is="<<ob.volume(r,h)<<"\n";

cout<<"Enter length,breadth and height of rectangle:";

cin>>l>>b>>h2;

cout<<"Volume of rectangle ="<<ob.volume(l,b,h2)<<"\n";

}

**OUTPUT:**

Ente side of a cube:4

Volume of a cube is=64

Enter radious and height of cylinder:3 5

Volume of cylinder is=141.3

Enter length,breadth and height of rectangle: 4 3 5

Volume of rectangle =60

**PROGRAM NO: 10**

**AIM: Create a 'MATRIX' class of size m X n. Overload the ‘+’ and ‘\*’ operator to add and multiply two MATRIX objects.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class matrix

{

public:

int a[50][50],i,j,k;

int r,c;

void read\_size()

{

cout<<"Enter row size and column size:";

cin>>r>>c;

}

void read()

{

cout<<"Enter values:";

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

{

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

{

cin>>a[i][j];

}

}

}

void display()

{

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

{

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

{

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

}

cout<<"\n";

}

}

matrix operator +(matrix);

matrix operator \*(matrix);

};

matrix matrix::operator +(matrix ob)

{

matrix temp;

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

{

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

{

temp.a[i][j]=0;

temp.a[i][j]=a[i][j]+ob.a[i][j];

}

temp.r=r;

temp.c=c;

}

return (temp);

}

matrix matrix::operator \*(matrix m)

{

matrix temp2;

temp2.r=r;

temp2.c=m.c;

for(i=0;i<temp2.r;i++)

{

for(j=0;j<temp2.c;j++)

{

temp2.a[i][j]=0;

for (int k=0;k<c;k++)

{

temp2.a[i][j]=temp2.a[i][j]+(a[i][k]\*m.a[k][j]);

}

}

}

return temp2;

}

main()

{

matrix m1,m2,m3,m4;

m1.read\_size();

m2.read\_size();

m1.read();

m2.read();

cout<<"First matrix is:\n";

m1.display();

cout<<"Second matrix is:\n";

m2.display();

if((m1.r==m2.r)&&(m1.c==m2.c))

{

m3=m1+m2;

cout<<"SUM IS:\n";

m3.display();

}

else

{

cout<<"Addition not possible\n";

}

if(m1.r==m2.c)

{

m4=m1\*m2;

cout<<"PRODUCT OF TWO MATRIX IS:\n";

m4.display();

}

else

{

cout<<"Multiplication Not possible\n";

}

}

**OUTPUT:**

Enter row size and column size:2 3

Enter row size and column size:2 3

Enter values:3 4 5 6 7 8

Enter values:3 4 5 7 8 0

First matrix is:

3 4 5

6 7 8

Second matrix is:

3 4 5

7 8 0

SUM IS:

6 8 10

13 15 8

Multiplication Not possible

Enter row size and column size:3 3

Enter row size and column size:3 3

Enter values:2 3 4 5 6 7 8 9 5

Enter values:2 3 4 1 7 2 3 8 10

First matrix is:

2 3 4

5 6 7

8 9 5

Second matrix is:

2 3 4

1 7 2

3 8 10

SUM IS:

4 6 8

6 13 9

11 17 15

PRODUCT OF TWO MATRIX IS:

19 59 54

37 113 102

40 127 100

Enter row size and column size:2 3

Enter row size and column size:3 2

Enter values:1 2 5 8 10 3

Enter values:12 4 5 7 9 9

First matrix is:

1 2 5

8 10 3

Second matrix is:

12 4

5 7

9 9

Addition not possible

PRODUCT OF TWO MATRIX IS:

67 63

173 129

**PROGRAM NO: 11**

**AIM: Write a program to implement inheritance.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class publisher

{

char title[20];

float price;

public:

publisher()

{

price=0;

title[20]=0;

}

void read()

{

cout<<"Enter title and price:";

cin>>title>>price;

}

void display()

{

cout<<"Title:"<<title<<"\n";

cout<<"Price :"<<price<<"\n";

}

};

class book:public publisher

{

int pagecount;

public:

book()

{

pagecount=0;

}

void bookread()

{

publisher::read();

cout<<"Enter number of pages:";

cin>>pagecount;

}

void bookdisplay()

{

cout<<".......DETAILS.........\n";

publisher::display();

cout<<"No. of pages:"<<pagecount<<"\n";

}

};

class cd:public publisher

{

int duration;

public:

cd()

{

duration=0;

}

void cdread()

{

publisher::read();

cout<<"Enter duration :";

cin>>duration;

}

void cddisplay()

{

publisher::display();

cout<<"Duration of cd:"<<duration<<"\n";

cout<<"\n";

}

};

main()

{

book b;

cd c;

b.bookread();

c.cdread();

b.bookdisplay();

c.cddisplay();

}

**OUTPUT:**

Enter title and price:Algorithms 350

Enter number of pages: 300

Enter title and price: c++

50

Enter duration :20m

.......DETAILS.........

Title:Algorithms

Price :350

No. of pages:300

Title:c++

Price :50

Duration of cd:20

**PROGRAM NO: 12**

**AIM: Write a program to implement virtual base class.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class base

{

public:

int i;

};

class derived1:virtualpublic base

{

public:

int j;

};

class derived2:virtualpublic base

{

public:

int k;

};

class derived3:public derived1,public derived2

{

public:

int sum;

};

main()

{

derived3 ob;

ob.i=10;

ob.j=20;

ob.k=34;

ob.sum=ob.i+ob.j+ob.k;

cout<<"Three numbers are:\n";

cout<<ob.i<<” from super class\n";

cout<<ob.j<<"from derived1 class\n";

cout<<ob.k<<"from derived2 class\n";

cout<<"SUM ="<<ob.sum<<"\n";

}

**OUTPUT:**

Three numbers are:

10 from super class

20 from derived1 class

34 from derived2 class

SUM =64

**PROGRAM NO: 13**

**AIM: Program to implement polymorphism.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class rectangle

{

float w,h,rect;

public:

void area()

{

cout<<"Enter width and height of rectangle:";

cin>>w>>h;

rect=w\*h;

cout<<"Area of rectangle="<<rect<<"\n";

}

};

class square

{

float a,sq;

public:

void area()

{

cout<<"Enter side of square:";

cin>>a;

sq=a\*a;

cout<<"Area of Square="<<sq<<"\n";

}

};

class circle

{

float r,circ;

public:

void area()

{

cout<<"Enter radious of circle:";

cin>>r;

circ=3.14\*r\*r;

cout<<"Area of circle="<<circ<<"\n";

}

};

main()

{

rectangle ob1 ;

square ob2;

circle ob3;

ob1.area();

ob2.area();

ob3.area();

}

**OUTPUT:**

Enter width and height of rectangle: 4 5

Area of rectangle=20

Enter side of square:2.3

Area of Square=5.29

Enter radious of circle:4.2

Area of circle=55.3896

**PROGRAM NO: 14**

**AIM: Program to implement virtual functions.**

**PROGRAM:**

#include<iostream>

usingnamespace std;

class base

{

public:

virtualvoid vfun()

{

cout<<"This is base's vfun()\n";

}

};

class derived1:public base

{

public:

void vfun()

{

cout<<"Dreived1's vfun()\n";

}

};

class derived2:public base

{

public:

void vfun()

{

cout<<"Dreived2's vfun()\n";

}

};

main()

{

base \*p,b;

derived1 d1;

derived2 d2;

p=&b;

p->vfun();

p=&d1;

p->vfun();

p=&d2;

p->vfun();

}

**OUTPUT:**

This is base's vfun()

Dreived1's vfun()

Dreived2's vfun()

**PROGRAM NO: 15**

**AIM: Write a program to implement C++ files.**

**PROGRAM:**

#include<iostream>

#include<fstream>

usingnamespace std;

class student

{

private:

int rollno,m1,m2,m3;

char name[20];

int total;

float avg;

public:

void getdata()

{

cout<<"Enter roll number\t"<<"\n";

cin>>rollno;

cout<<"Enter student name\t"<<"\n";

cin>>name;

cout<<"Enter marks of three subjects\t"<<"\n";

cin>>m1>>m2>>m3;

}

void putdata()

{

cout<<"Roll No.:"<<rollno<<"\n";

cout<<"Name :"<<name<<"\n";

cout<<"marks:"<<m1<<" "<<m2<<" "<<m3<<"\n";

total=m1+m2+m3;

avg=total/3;

cout<<"TotalMark :"<<total<<"\n";

cout<<"Average:"<<avg<<"\n";

}

};

main()

{

cout<<"C++ FILES \n------------------";

cout<<"\n";

student st;

st.getdata();

ofstream outfile("mark.out");

outfile.write((char\*)&st,sizeof(st));

outfile.close();

cout<<"\*\*\*Display file\*\*\*\n";

ifstream infile("mark.out");

infile.read((char\*)&st,sizeof(st));

st.putdata();

}

**OUTPUT:**

C++ FILES

------------------

Enter roll number

14

Enter student name

Safna

Enter marks of three subjects

45

56

43

\*\*\*Display file\*\*\*

Roll No.:14

Name :Safna

marks:45 56 43

TotalMark :144

Average:48

**PROGRAM NO: 16**

**AIM: Write a program to demonstrate IO based program.**

**PROGRAM:**

#include<iostream>

#include<cstring>

usingnamespace std;

class phonebook

{

char name[20];

int areacode,prefix;

longint num;

public:

phonebook()

{

}

phonebook(char\*n,int a,int p,int nm)

{

strcpy(name,n);

areacode=a;

prefix=p;

num=nm;

}

friend ostream &operator<<(ostream &stream,phonebook o);

friend istream &operator>>(istream &stream,phonebook &o);

};

ostream &operator<<(ostream &stream,phonebook o)

{

stream<<o.name<<" ";

stream<<"("<<o.areacode<<")";

stream<<o.prefix<<"-"<<o.num<<"\n";

return stream;

}

istream &operator>>(istream &stream,phonebook &o)

{

cout<<"Enter name:";

stream>>o.name;

cout<<"Enter areacode:";

stream>>o.areacode;

cout<<"Enter prefix:";

stream>>o.prefix;

cout<<"Enter number:";

stream>>o.num;

cout<<"\n";

return stream;

}

main()

{

phonebook ob;

cin>>ob;

cout<<ob;

}

**OUTPUT:**

Enter name:Aleesha

Enter areacode:706

Enter prefix:91

Enter number:8113854402

Aleesha (706)91-8113854402

**PROGRAM NO: 17**

**AIM: Write a program to demonstrate IO based program.**

**PROGRAM:**

#include<iostream>

#include<strstream>

usingnamespace std;

main()

{

char iostr[80];

strstream strio(iostr,sizeof(iostr),ios::in|ios::out);

int a,b;

char str[80];

strio<<"10 20 testing";

strio>>a>>b>>str;

cout<<a<<" "<<b<<" "<<str<<endl;

}

**OUTPUT:**

10 20 testing

**PROGRAM NO: 18**

**AIM: Program to demonstrate Standard Template Library(STL).**

**PROGRAM:**

#include<iostream>

#include<vector>

#include<cctype>

usingnamespace std;

main()

{

vector<char>v(10);

unsignedint i;

cout<<"Size="<<v.size()<<endl;

for(i=0;i<10;i++)

v[i]=i+'a';

cout<<"Current Elements\n";

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

cout<<v[i]<<" ";

cout<<"\n\n";

cout<<"Expanding Vector:\n";

for(i=0;i<10;i++)

v.push\_back(i+10+'a');

cout<<"size new ="<<v.size()<<endl;

cout<<"Current contents\n";

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

cout<<v[i]<<" ";

cout<<"\n\n";

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

v[i]=toupper(v[i]);

cout<<"Modified Contents\n";

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

cout<<v[i]<<" ";

cout<<"\n\n";

}

**OUTPUT:**

Size=10

Current Elements

a b c d e f g h i j

Expanding Vector:

size new =20

Current contents

a b c d e f g h i j k l m n o p q r s t

Modified Contents

A B C D E F G H I J K L M N O P Q R S T

**DATA STRUCTURE**

**PROGRAM NO: 1**

**AIM: Program to implement singly linked list operations.**

**PROGRAM:**

#include<iostream>

using namespace std;

class single

{

private:

struct node

{ int data;

node \*link;

}\*p;

public:

single();

void append(int);

void addbeg(int);

void addafter(int,int);

void display();

int count();

void del(int);

~single();

};

single::single()

{

p=NULL;

}

void single::append(int num)

{

node \*temp,\*r;

temp=new node;

r=new node;

if(p==NULL)

{

temp=new node;

temp->data=num;

temp->link=NULL;

p=temp;

}

else

{

for(temp=p;temp->link!=NULL;temp=temp->link);

{

r->data=num;

r->link-NULL;

temp->link=r;

}

}

}

void single::addbeg(int num)

{

node \*temp;

temp=new node;

if(p==NULL)

{

temp->data=num;

temp->link=NULL;

p=temp;

}

else

{

temp->data=num;

temp->link=p;

p=temp;

}

}

void single::addafter(int loc,int num)

{

int i;

node \*temp,\*r;

temp=p;

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

{

temp=temp->link;

if(temp==NULL)

{

cout<<"Less thaan"<<loc<<"elements\n";

}

}

r=new node;

r->data=num;

r->link=temp->link;

temp->link=r;

}

void single::del(int num)

{

node \*old,\*temp;

temp=p;

while(temp!=NULL)

{

if(temp->data==num)

{

if(temp==p)

p=temp->link;

else

old->link=temp->link;

delete temp;

return;

}

else

{

old=temp;

temp=temp->link;

}

}

cout<<"NOT FOUND\n";

}

int single::count()

{

node \*temp=p;

int c=0;

while(temp!=NULL)

{

temp=temp->link;

c++;

}

return c;

}

void single::display()

{

node \*temp;

for(temp=p;temp->link!=NULL;temp=temp->link)

{

cout<<temp->data<<"->";

}

cout<<temp->data;

}

single::~single()

{

node \*q;

while(q!=NULL)

{

q=q->link;

delete p;

p=q;

}

}

main()

{

single ob;

int a,n,ch,loc,m,b,d;

cout<<"SINGLY LINKED LIST";

do{

cout<<"\n1.ADD BEG\n2.APPEND\n3.ADD AFTER\n4.DELETE\n5.EXIT\n";

cout<<"Enter choice\n";

cin>>ch;

switch(ch)

{

case 1:cout<<"Enter howmany number of elements to be added first:";

cin>>n;

cout<<"Enter the element:";

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

{

cin>>a;

ob.addbeg(a);

}

cout<<"Elements are :";

ob.display();

break;

case 2:cout<<"Enter how many elements to be added last:";

cin>>n;

cout<<"Enter elements:";

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

{

cin>>a;

ob.append(a);

}

cout<<"Nummber of elements:"<<ob.count()<<"\n";

cout<<"Elements are:";

ob.display();

break;

case 3:cout<<"Enter number of elements to be inserted:";

cin>>n;

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

{

cout<<"Enter position: ";

cin>>loc;

cout<<"Enter the element:\n";

cin>>a;

ob.addafter(loc,a);

}

cout<<"Number of elements are:"<<ob.count();

cout<<"\n";

cout<<"Elements are:";

ob.display();

break;

case 4:cout<<"Enter the element to be deleted \n";

cin>>d;

ob.del(d);

cout<<"\n";

cout<<"Elements are:";

ob.display();

break;

case 5:cout<<"EXIT\n";

break;

}

}while(ch!=5);

}

**OUTPUT:**

SINGLY LINKED LIST

1.ADD BEG

2.APPEND

3.ADD AFTER

4.DELETE

5.EXIT

Enter choice

1

Enter how many number of elements to be added first:2

Enter the element:20 10

Elements are :10->20

1.ADD BEG

2.APPEND

3.ADD AFTER

4.DELETE

5.EXIT

Enter choice

2

Enter how many elements to be added last:3

Enter elements:40 50 70

Nummber of elements:5

Elements are:10->20->40->50->70

1.ADD BEG

2.APPEND

3.ADD AFTER

4.DELETE

5.EXIT

Enter choice

3

Enter number of elements to be inserted:2

Enter position: 1

Enter the element:

30

Enter position: 4

Enter the element:

60

Number of elements are:7

Elements are:10->20->30->40->50->60->70

1.ADD BEG

2.APPEND

3.ADD AFTER

4.DELETE

5.EXIT

Enter choice

4

Enter the element to be deleted

70

Elements are:10->20->30->40->50->60

1.ADD BEG

2.APPEND

3.ADD AFTER

4.DELETE

5.EXIT

Enter choice

4

Enter the element to be deleted

30

Elements are:10->20->40->50->60

1.ADD BEG

2.APPEND

3.ADD AFTER

4.DELETE

5.EXIT

Enter choice

5

EXIT

**PROGRAM NO: 2**

**AIM: Write a program to implement ascending order sorting in SLL.**

**PROGRAM:**

#include<iostream>

using namespace std;

class link

{

private:

struct node

{

int data;

node \*link;

}\*p;

public:

link();

void append(int num);

void display();

void sort(int);

~link();

}ob;

link::link()

{

p=NULL;

}

void link::append(int num)

{

node \*temp,\*r;

temp=new node;

r=new node;

if(p==NULL)

{

temp=new node;

temp->data=num;

temp->link=NULL;

p=temp;

}

else

{

for(temp=p;temp->link!=NULL;temp=temp->link);

{

r->data=num;

r->link-NULL;

temp->link=r;

}

}

}

void link::sort(int n)

{

int temp;

node \*q,\*r;

q=p;

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

{

r=q->link;

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

{

if(q->data>r->data)

{

temp=q->data;

q->data=r->data;

r->data=temp;

}

r=r->link;

}

q=q->link;

}

}

void link::display()

{

node \*temp;

for(temp=p;temp->link!=NULL;temp=temp->link)

{

cout<<temp->data<<"->";

}

cout<<temp->data;

}

link::~link()

{

node \*q;

while(p!=NULL)

{

q=p->link;

delete p;

p=q;

} }

main()

{

int n,a;

cout<<"Enter limit:";

cin>>n;

cout<<"Enter elements:";

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

{

cin>>a;

ob.append(a);

}

cout<<"Elements are:";

ob.display();

cout<<"\n";

cout<<"Linked list after sorting:";

ob.sort(n);

ob.display();

cout<<"\n";

}

**OUTPUT:**

Enter limit:5

Enter elements:50 30 10 20 45

Elements are:50->30->10->20->45

Linked list after sorting:10->20->30->45->50

**PROGRAM NO: 3**

**AIM: Write a program to find reverse order of a SLL.**

**PROGRAM:**

#include<iostream>

using namespace std;

class link

{

private:

struct node

{

int data;

node \*link;

}\*p;

public:

link();

void addbeg(int num);

void reverse();

void display();

int count();

~link();

}ob;

link::link()

{

p=NULL;

}

void link::addbeg(int num)

{

node \*temp;

temp=new node;

if(p==NULL)

{

temp->data=num;

temp->link=NULL;

p=temp;

}

else

{

temp->data=num;

temp->link=p;

p=temp;

}

}

void link::reverse()

{

node \*q,\*s,\*r;

q=p;

r=NULL;

while(q!=NULL)

{

s=r;

r=q;

q=q->link;

r->link=s;

}

p=r;

}

void link::display()

{

node \*temp;

for(temp=p;temp->link!=NULL;temp=temp->link)

{

cout<<temp->data<<"->";

}

cout<<temp->data;

}

int link::count()

{

node \*temp=p;

int c=0;

while(temp!=NULL)

{

temp=temp->link;

c++;

}

return c;

}

link::~link()

{

node \*q;

while(p!=NULL)

{

q=p->link;

delete p;

p=q;

}

}

main()

{

int n,a;

cout<<"Enter limit:";

cin>>n;

cout<<"Enter the element:";

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

{

cin>>a;

ob.addbeg(a);

}

cout<<"Elements are:";

ob.display();

cout<<"\n";

cout<<"Reversed elements are:";

ob.reverse();

ob.display();

cout<<"\n";

}

**OUTPUT:**

Enter limit:5

Enter the element:34 2 76 3 8

Elements are:8->3->76->2->34

Reversed elements are:34->2->76->3->8

**PROGRAM NO:4**

**AIM: Write a program to implement doubly linked list operations.**

**PROGRAM:**

#include<iostream>

using namespace std;

class doubly

{

public:

int item;

struct node

{

node \*prev;

node \*next;

int data;

}\*p;

doubly()

{

p=new node;

p=NULL;

}

void append();

void instbeg();

void instbw();

void delt(int);

void display();

};

void doubly::append()

{

node \*temp,\*r;

temp=new node;

r=new node;

cout<<"enter item:";

cin>>item;

temp->data=item;

temp->next=NULL;

if(p==NULL)

{

temp->prev=NULL;

p=temp;

}

else

{

for(r=p;r->next!=NULL;r=r->next);

r->next=temp;

temp->prev=r;

}

}

void doubly::instbeg()

{

node \*temp;

temp=new node;

cout<<"enter item:";

cin>>item;

temp->data=item;

temp->prev=NULL;

if(p==NULL)

{

temp->next=NULL;

}

else

{

temp->next=p;

p->prev=temp;

}

p=temp;

}

void doubly::instbw()

{

node \*temp,\*r;

temp=new node;

r=new node;

int item,l,i;

cout<<"Enter item:";

cin>>item;

cout<<"Enter location:";

cin>>l;

if(p==NULL)

cout<<"List empty\n";

else

{

r=p;

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

{

r=r->next;

if(r==NULL)

{

cout<<"Lesser number of nodes\n";

return;

}

}

if(r->next!=NULL)

{

(r->next)->prev=temp;

}

temp->next=r->next;

temp->data=item;

r->next=temp;

temp->prev=r;

}

}

void doubly::delt(int item)

{

node \*temp=p;

while(temp!=NULL)

{

if(temp->data==item)

{

if(temp==p)

{

p=p->next;

p->prev=NULL;

}

else

{

if(temp->next==NULL)

{

temp->prev->next=NULL;

}

else

{

(temp->prev)->next=temp->next;

(temp->next)->prev=temp->prev;

}

delete temp;

}

cout<<item<<" "<<" element deleted\n";

return;

}

temp=temp->next;

}

cout<<item<<" "<<"Not found\n";

}

void doubly::display()

{

node \*temp;

for(temp=p;temp->next!=NULL;temp=temp->next)

cout<<temp->data<<"->";

cout<<temp->data<<"\n";

}

main()

{

doubly ob;

int ch,n,d;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DOUBLY LINKED LIST\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout<<"1.INSERT FRONT\n2.APPEND\n3.INSERT B/W\n4.DELETE\n5.DISPLAY\n6.EXIT\n";

do

{

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:cout<<"Enter how many number of elements:";

cin>>n;

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

{

ob.instbeg();

}

break;

case 2:cout<<"Enter how many number of elements:";

cin>>n;

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

{

ob.append();

}

break;

case 3:cout<<"Enter how many number of elements:";

cin>>n;

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

{

ob.instbw();

}

break;

case 4:cout<<"Enter the item to be deleted:";

cin>>d;

ob.delt(d);

break;

case 5:ob.display();

break;

case 6:cout<<"EXITT!!\n";

break;

}

}while(ch!=6);

}

**OUTPUT:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DOUBLY LINKED LIST\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1.INSERT FRONT

2.APPEND

3.INSERT B/W

4.DELETE

5.DISPLAY

6.EXIT

Enter your choice:1

Enter how many number of elements:2

enter item:10

enter item:20

Enter your choice:2

Enter how many number of elements:3

enter item:30

enter item:40

enter item:50

Enter your choice:5

20->10->30->40->50

Enter your choice:3

Enter how many number of elements:2

Enter item:25

Enter location:1

Enter item:34

Enter location:3

Enter your choice:5

20->10->25->30->34->40->50

Enter your choice:4

Enter the item to be deleted:55

55 Not found

Enter your choice:4

Enter the item to be deleted:30

30 element deleted

Enter your choice:4

Enter the item to be deleted:34

34 element deleted

Enter your choice:5

20->10->25->40->50

Enter your choice:6

EXITT!!

**PROGRAM NO:5**

**AIM: Write a program to implement stack using singly linked list.**

**PROGRAM:**

#include<iostream>

using namespace std;

class stack

{

public:

int n,item;

struct node

{

int data;

node \*link;

}\*top;

stack()

{

top=NULL;

}

void push(int);

void pop();

void display();

};

void stack::push(int item)

{

node \*temp;

temp=new node;

cout<<"Enter Item:";

cin>>item;

temp->data=item;

temp->link=top;

cout<<temp->data<<" "<<"pushed \n";

top=temp;

}

void stack::pop()

{

node \*temp;

temp=new node;

if(top==NULL)

{

cout<<"STACK EMPTY\n";

}

else

{

cout<<top->data<<" "<<"poped\n";

delete top;

top=top->link;

}

}

void stack::display()

{

node \*r;

r=new node;

if(top==NULL)

{

cout<<"STACK EMPTY\n";

}

else

{

for(r=top;r->link!=NULL;r=r->link)

{

cout<<r->data<<"->";

}

cout<<r->data<<"\n";

}

}

main()

{

stack s;

int c,n,item,top;

cout<<"\*\*\*\*\*\*STACK\*\*\*\*\*\*\n";

cout<<"1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n";

do

{

cout<<"Enter your choice:";

cin>>c;

switch(c)

{

case 1:s.push(item);

break;

case 2:s.pop();

break;

case 3:s.display();

break;

case 4:cout<<"EXIT!!!!\n";

break;

}

}while(c!=4);

}

**OUTPUT:**

\*\*\*\*\*\*STACK\*\*\*\*\*\*

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter your choice:1

Enter Item:2

2 pushed

Enter your choice:1

Enter Item:3

3 pushed

Enter your choice:1

Enter Item:5

5 pushed

Enter your choice:3

5->3->2

Enter your choice:2

5 poped

Enter your choice:2

3 poped

Enter your choice:2

2 poped

Enter your choice:3

STACK EMPTY

Enter your choice:2

STACK EMPTY

Enter your choice:4

EXIT!!!!

**PROGRAM NO:6**

**AIM: Write a program to implement queue using array.**

**PROGRAM:**

#include<iostream>

using namespace std;

class queue

{

int n,front,rear,item,i,q[10];

public:

queue()

{

front=0;

rear=0;

};

void read();

void insert(int);

void delt();

void display();

};

void queue::read()

{

cout<<"Enter size:";

cin>>n;

}

void queue::insert(int item)

{

if(rear==n)

{

cout<<"QUEUE IS FULL\n";

}

else

{

cout<<"Enter element:";

cin>>item;

rear=rear+1;

q[rear]=item;

cout<<item<<" "<<"Inserted\n";

}

if((front==0)&&(rear>0))

{

front=1;

}

}

void queue::delt()

{

if(front>rear)

{

front=0;

rear=0;

cout<<"QUEUE EMPTY\n";

}

else

{

if(front!=0)

{

item=q[front];

cout<<item<<" "<<"deleted\n";

front=front+1;

}

else

{

cout<<"QUEUE EMPTY\n";

}

}

}

void queue::display()

{

int i;

if(front>rear)

{

cout<<"QUEUE EMPTY\n";

}

else

{

cout<<"Queue elements are:\n";

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

{

cout<<q[i]<<"\n";

}

}

}

main()

{

queue ob;

int ch,n,item;

ob.read();

cout<<"\*\*\*\*\*QUEUE\*\*\*\*\*\n";

cout<<"1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n";

do

{

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:ob.insert(item);

break;

case 2:ob.delt();

break;

case 3:ob.display();

break;

case 4:cout<<"EXITT!!!!\n";

break;

}

}while(ch!=4);

}

**OUTPUT:**

Enter size:5

\*\*\*\*\*QUEUE\*\*\*\*\*

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice:1

Enter element:4

4 Inserted

Enter your choice:1

Enter element:8

8 Inserted

Enter your choice:1

Enter element:6

6 Inserted

Enter your choice:1

Enter element:7

7 Inserted

Enter your choice:1

Enter element:6

6 Inserted

Enter your choice:1

QUEUE IS FULL

Enter your choice:3

Queue elements are:

4

8

6

7

6

Enter your choice:2

4 deleted

Enter your choice:2

8 deleted

Enter your choice:2

6 deleted

Enter your choice:2

7 deleted

Enter your choice:2

6 deleted

Enter your choice:2

QUEUE EMPTY

Enter your choice:4

EXITT!!!!

**PROGRAM NO:7**

**AIM: Write a program to convert infix expression to postfix.**

**PROGRAM:**

#include<iostream>

#include<string.h>

#include<ctype.h>

using namespace std;

const int MAX=50;

class infix

{

private:

char target[MAX],stack[MAX];

char \*s,\*t;

int top;

public:

infix();

void setexpr(char \*str);

void push(char c);

char pop();

void convert();

int priority(char c);

void show();

};

infix::infix()

{

top=-1;

strcpy(target,"");

strcpy(stack,"");

t=target;

s="";

}

void infix::setexpr(char \*str)

{

s=str;

}

void infix::push(char c)

{

if(top==MAX)

cout<<"STACK FULL\n";

else

{

top++;

stack[top]=c;

}

}

char infix::pop()

{

if(top==-1)

{

cout<<"STACK EMPTY\n";

return 1;

}

else

{

char item=stack[top];

top--;

return item;

}

}

void infix::convert()

{

while(\*s)

{

if(\*s==' '||\*s=='\t')

{

s++;

continue;

}

if(isdigit(\*s)||isalpha(\*s))

{

while(isdigit(\*s)||isalpha(\*s))

{

\*t=\*s;

s++;

t++;

}

}

if(\*s=='(')

{

push(\*s);

s++;

}

char opr;

if(\*s=='\*'||\*s=='+'||\*s=='/'||\*s=='%'||\*s=='-'||\*s=='$')

{

if(top!=-1)

{

opr=pop();

while(priority(opr)>=priority(\*s))

{

\*t=opr;

t++;

opr=pop();

}

push(opr);

push(\*s);

}

else

push(\*s);

s++;

}

if(\*s==')')

{

opr=pop();

while((opr)!='(')

{

\*t=opr;

t++;

opr=pop();

}

s++;

}

}

while(top!=-1)

{

char opr=pop();

\*t=opr;

t++;

}

\*t='\0';

}

int infix::priority(char c)

{

if(c=='$')

return 3;

if(c=='\*'||c=='/'||c=='%')

return 2;

else

{

if(c=='+'||c=='-')

return 1;

else

return 0;

}

}

void infix::show()

{

cout<<target;

}

main()

{

char expr[MAX];

infix q;

cout<<"Enter an infix expression:";

cin.getline(expr,MAX);

q.setexpr(expr);

q.show();

q.convert();

cout<<"The postfix Expression is:";

q.show();

}

**OUTPUT:**

Enter an infix expression:34+5\*6

The postfix Expression is:3456\*+

Enter an infix expression:(3+4)\*(45+5)

The postfix Expression is:34+455+\*

Enter an infix expression:2+3+5+8+7

STACK EMPTY

The postfix Expression is:23+5+8+7+

**PROGRAM NO: 8**

**AIM: Write a program to implement postfix expression evaluation.**

**PROGRAM:**

#include<iostream>

#include<string.h>

#include<math.h>

using namespace std;

class eval

{

int top,n,a[10];

char str[20];

public:

eval()

{

top=0;

}

void read();

void push(int);

int pop();

void calc();

void display();

};

void eval::read()

{

cout<<"Enter size:";

cin>>n;

cout<<"Enter the postfix expression:";

cin>>str;

}

void eval::push(int c)

{

if(top==n)

cout<<"STACK FULL\n";else

{

top++;

a[top]=c;

cout<<c<<" "<<"inserted\n";

}

}

int eval::pop()

{

int j;

if(top==0)

{

cout<<"STACK EMPTY\n";

}

else

{

j=a[top];

cout<<j<<" "<<"is poped from the stack\n";

top--;

}

return j;

}

void eval::calc()

{

int i,m1,m2,m3,l;

l=strlen(str);

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

{

if(str[i]==' ')

{

continue;

}

if(isdigit(str[i]))

{

int x=str[i]-'0';

push(x);

}

if(str[i]=='+'||str[i]=='-'||str[i]=='\*'||str[i]=='/')

{

cout<<str[i]<<" "<<"operator\n";

m2=pop();

m1=pop();

switch(str[i])

{

case '+':m3=m1+m2;

break;

case '-':m3=m1-m2;

break;

case '\*':m3=m1\*m2;

break;

case '/':m3=m1/m2;

break;

default:continue;

}

push(m3);

}

}

}

void eval::display()

{

cout<<"The result of the postfix expression=";

cout<<a[top];

cout<<"\n";

}

main()

{

eval ob;

cout<<"\_\_\_\_\_\_POSTFIX EVALUATION\_\_\_\_\_\_\n";

ob.read();

ob.calc();

ob.display();

}

**OUTPUT:**

\_\_\_\_\_\_POSTFIX EVALUATION\_\_\_\_\_\_

Enter size:5

Enter the postfix expression:34-567/\*+

3 inserted

4 inserted

- operator

4 is poped from the stack

3 is poped from the stack

-1 inserted

5 inserted

6 inserted

7 inserted

/ operator

7 is poped from the stack

6 is poped from the stack

0 inserted

\* operator

0 is poped from the stack

5 is poped from the stack

0 inserted

+ operator

0 is poped from the stack

-1 is poped from the stack

-1 inserted

The result of the postfix expression=-1

**PROGRAM NO: 9**

**AIM: Write a program to implement circular queue operations.**

**PROGRAM:**

#include<iostream>

using namespace std;

class cqueue

{

public:

int front,rear,n,q[20];

cqueue()

{

front=0;

rear=0;

}

void insert();

void delt();

void display();

};

void cqueue::insert()

{

int item;

if((rear==n&&front==1)||(rear+1==front))

{

cout<<"QUEUE FULL\n";

}

else

{

cout<<"Enter item:";

cin>>item;

rear=rear+1;

if(rear==n+1)

rear=1;

q[rear]=item;

cout<<item<<" "<<"Inserted\n";

if(front==0)

front=1;

}

}

void cqueue::delt()

{

int item;

if(front==0)

{

cout<<"QUEUE EMPTY\n";

}

else

{

if(front!=rear)

{

item=q[front];

cout<<item<<" "<<"deleted\n";

front++;

if(front==n+1)

front=1;

}

else

{

item=q[front];

cout<<item<<" "<<"Deleted\n";

front=0;

rear=0;

}

}

}

void cqueue::display()

{

int i;

if(front==0)

{

cout<<"QUEUE EMPTY\n";

}

else

{

cout<<"Queue elements are:";

for(i=front;i!=rear;)

{

cout<<q[i]<<" ";

if(i==n)

i=1;

else

i++;

}

cout<<q[rear]<<"\n";

}

}

main()

{

cqueue ob;

cout<<"Enter size:";

cin>>ob.n;

int ch;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_CIRCULAR QUEUE\_\_\_\_\_\_\_\_\_\n";

cout<<"1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n";

do

{

cout<<"Enter your choice:";

cin>>ch;

switch(ch)

{

case 1:ob.insert();

break;

case 2:ob.delt();

break;

case 3:ob.display();

break;

case 4:cout<<"EXIT!!!\n";

break;

}

}while(ch!=4);

}

**OUTPUT:**

Enter size:8

\_\_\_\_\_\_\_\_\_\_\_\_CIRCULAR QUEUE\_\_\_\_\_\_\_\_\_

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice:1

Enter item:0

0 Inserted

Enter your choice:1

Enter item:1

1 Inserted

Enter your choice:1

Enter item:2

2 Inserted

Enter your choice:1

Enter item:3

3 Inserted

Enter your choice:2

0 deleted

Enter your choice:2

1 deleted

Enter your choice:3

Queue elements are:2 3

Enter your choice:1

Enter item:4

4 Inserted

Enter your choice:1

Enter item:5

5 Inserted

Enter your choice:1

Enter item:6

6 Inserted

Enter your choice:1

Enter item:7

7 Inserted

Enter your choice:1

Enter item:8

8 Inserted

Enter your choice:3

Queue elements are:2 3 4 5 6 7 8

Enter your choice:1

Enter item:9

9 Inserted

Enter your choice:1

QUEUE FULL

Enter your choice:2

2 deleted

Enter your choice:2

3 deleted

Enter your choice:3

Queue elements are:4 5 6 7 8 9

Enter your choice:2

4 deleted

Enter your choice:2

5 deleted

Enter your choice:2

6 deleted

Enter your choice:2

7 deleted

Enter your choice:2

8 deleted

Enter your choice:2

9 Deleted

Enter your choice:2

QUEUE EMPTY

Enter your choice:4

EXIT!!!

**PROGRAM NO: 10**

**AIM: Write a program to implement binary tree operations.**

**PROGRAM:**

#include<iostream>

using namespace std;

struct node

{

node\* left;

char data;

node \*right;

};

class btree

{

private:

node\* root;

char\* arr;

int\* lc;

int\* rc;

int size;

public:

btree(char \*a,int \*l,int \*r,int size);

void insert(int index);

static node \*buildtree(char \*a,int \*l,int \*r,int index);

void display();

static void inorder(node \*sr);

~btree();

static void del(node \*sr);

};

btree::btree(char \*a,int \*l,int \*r,int size)

{

root=NULL;

arr=new char[size];

lc=new int[size];

rc=new int[size];

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

{

\*(arr+i)=\*(a+i);

\*(lc+i)=\*(l+i);

\*(rc+i)=\*(r+i);

}

}

void btree::insert(int index)

{

root=buildtree(arr,lc,rc,index);

}

node\* btree::buildtree(char \*a,int \*l,int \*r,int index)

{

node\* temp=NULL;

if(index!=-1)

{

temp=new node;

temp->left=buildtree(a,l,r,\*(l+index));

temp->data=\*(a+index);

temp->right=buildtree(a,l,r,\*(r+index));

}

return temp;

}

void btree::display()

{

inorder(root);

}

void btree::inorder(node \*sr)

{

if(sr!=NULL)

{

inorder(sr->left);

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

inorder(sr->right);

}

}

btree::~btree()

{

delete arr;

delete lc;

delete rc;

del(root);

}

void btree::del(node\* sr)

{

if(sr!=NULL)

{

del(sr->left);

del(sr->right);

}

delete sr;

}

main()

{

cout<<"Creating binary tree\n";

char a[]={'A','B','C','D','E','F','G','\0','\0','H'};

int l[]={1,3,5,-1,9,-1,-1,-1,-1,-1};

int r[]={2,4,6,-1,-1,-1,-1,-1,-1,-1};

int sz=sizeof(a);

btree bt(a,l,r,sz);

bt.insert(0);

cout<<"Inorder traversal of binary tree\n";

bt.display();

}

**OUTPUT:**

Creating binary tree

Inorder traversal of binary tree

D B H E A F C G

**PROGRAM NO: 11**

**AIM: Write a program to implement recursive traversal of binary tree.**

**PROGRAM:**

#include<iostream>

using namespace std;

class btree

{

private:

struct btreenode

{

btreenode \*leftchild;

int data;

btreenode \*rightchild;

}\*root;

public:

btree();

void buildtree(int num);

static void insert(btreenode \*\*sr,int num);

void traverse();

static void inorder(btreenode \*sr);

static void preorder(btreenode \*sr);

static void postorder(btreenode \*sr);

~btree();

static void del(btreenode \*sr);

};

btree::btree()

{

root=NULL;

}

void btree::buildtree(int num)

{

insert(&root,num);

}

void btree::insert(btreenode \*\*sr,int num)

{

if(\*sr==NULL)

{

\*sr=new btreenode;

(\*sr)->leftchild=NULL;

(\*sr)->data=num;

(\*sr)->rightchild=NULL;

return;

}

else

{

if(num<(\*sr)->data)

insert(&((\*sr)->leftchild),num);

else

insert(&((\*sr)->rightchild),num);

}

return;

}

void btree::traverse()

{

cout<<"Inorder Traversal:";

inorder(root);

cout<<"\nPreorder Traversal:";

preorder(root);

cout<<"\nPostorder Traversal:";

postorder(root);

cout<<"\n";

}

void btree::inorder(btreenode \*sr)

{

if(sr!=NULL)

{

inorder(sr->leftchild);

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

inorder(sr->rightchild);

}

else

return;

}

void btree::preorder(btreenode \*sr)

{

if(sr!=NULL)

{

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

preorder(sr->leftchild);

preorder(sr->rightchild);

}

else

return;

}

void btree::postorder(btreenode \*sr)

{

if(sr!=NULL)

{

postorder(sr->leftchild);

postorder(sr->rightchild);

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

}

else

return;

}

btree::~btree()

{

del(root);

}

void btree::del(btreenode \*sr)

{

if(sr!=NULL)

{

del(sr->leftchild);

del(sr->rightchild);

}

delete sr;

}

main()

{

cout<<"\_\_\_\_\_\_\_Recursive traversal of BinaryTree\_\_\_\_\_\_\n";

btree bt;

int req,i=1,num;

cout<<"Enter the number of items to be inserted:";

cin>>req;

while(i++<=req)

{

cout<<"Enter data:";

cin>>num;

bt.buildtree(num);

}

bt.traverse();

}

**OUTPUT:**

\_\_\_\_\_\_\_Recursive traversal of BinaryTree\_\_\_\_\_\_

Enter the number of items to be inserted:5

Enter data:6

Enter data:23

Enter data:1

Enter data:89

Enter data:5

Inorder Traversal: 1 5 6 23 89

Preorder Traversal: 6 1 5 23 89

Postorder Traversal: 5 1 89 23 6

**PROGRAM NO: 12**

**AIM: Write a program to implement non recursive traversal of binary tree.**

**PROGRAM:**

#include<iostream>

#include<stdlib.h>

using namespace std;

class node

{

public:

class node \*left;

class node \*right;

int data;

};

class tree:public node

{

public:

int stk[50],top;

node \*root;

tree()

{

root=NULL;

top=0;

}

void insert(int ch)

{

node \*temp,\*temp1;

if(root==NULL)

{

root=new node;

root->data=ch;

root->left=NULL;

root->right=NULL;

return;

}

temp1=new node;

temp1->data=ch;

temp1->right=temp1->left=NULL;

temp=search(root,ch);

if(temp->data>ch)

temp->left=temp1;

else

temp->right=temp1;

}

node \*search(node \*temp,int ch)

{

if(root==NULL)

{

cout<<"\n No node present";

return NULL;

}

if(temp->left==NULL&&temp->right==NULL)

return temp;

if(temp->data>ch)

{

if(temp->left==NULL)

return temp;

search(temp->left,ch);

}

else

{

if(temp->right==NULL)

return temp;

search(temp->right,ch);

}

}

void display(node \*temp)

{

if(temp==NULL)

return;

display(temp->left);

cout<<temp->data<<" ";

display(temp->right);

}

void preorder(node \*root)

{

node \*p,\*q;

p=root;

q=NULL;

top=0;

while(p!=NULL)

{

cout<<p->data<<" ";

if(p->right!=NULL)

{

stk[top]=p->right->data;

top++;

}

p=p->left;

if(p==NULL&&top>0)

{

p=pop(root);

}

}

}

void postorder(node \*root)

{

node \*p;

p=root;

top=0;

while(1)

{

while(p!=NULL)

{

stk[top]=p->data;

top++;

if(p->right!=NULL)

stk[top++]=-p->right->data;

p=p->left;

}

while(stk[top - 1]>0||top==0)

{

if(top==0)

return;

cout<<stk[top - 1]<<" ";

p=pop(root);

}

if(stk[top - 1]<0)

{

stk[top - 1]=-stk[top - 1];

p=pop(root);

}

}

}

void inorder(node \*root)

{

node \*p;

p=root;

top=0;

do

{

while(p!=NULL)

{

stk[top]=p->data;

top++;

p=p->left;

}

if(top>0)

{

p=pop(root);

cout<<p->data<<" " ;

p=p->right;

}

}

while(top!=0||p!=NULL);

}

node \*pop(node \*p)

{

int ch;

ch=stk[top - 1];

if(p->data==ch)

{

top--;

return p;

}

if(p->data>ch)

pop(p->left);

else

pop(p->right);

}

};

int main(int argc,char \*\*argv)

{

tree t1;

int ch,c,i,n;

node \*temp;

do

{

cout<<"\n1.INSERT\n2.DISPLAY\n3.INORDER TRAVERSAL\n4.PREORDER TRAVERSAL\n5.POSTORDER TRAVERSAL\n6.EXIT\n";

cout<<"ENTER YOUR CHOICE:";

cin>>ch;

switch(ch)

{

case 1:cout<<"Enter the number of elements to be inserted:";

cin>>n;

cout<<"Enter the elements:";

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

{

cin>>ch;

t1.insert(ch);

}

break;

case 2:cout<<"The elements are:";

t1.display(t1.root);

break;

case 3:cout<<"After inorder traversal\n";

t1.inorder(t1.root);

break;

case 4:cout<<"After preorder traversal\n";

t1.preorder(t1.root);

break;

case 5:cout<<"After postorder traversal\n";

t1.postorder(t1.root);

break;

case 6:exit(1);

}

cout<<"\n";

cout<<"Do you want to continue(1/0):";

cin>>c;

}while(c==1);

}

**OUTPUT:**

1.INSERT

2.DISPLAY

3.INORDER TRAVERSAL

4.PREORDER TRAVERSAL

5.POSTORDER TRAVERSAL

6.EXIT

ENTER YOUR CHOICE:1

Enter the number of elements to be inserted:10

Enter the elements:11 2 9 13 57 25 17 1 90 3

Do you want to continue(1/0):1

1.INSERT

2.DISPLAY

3.INORDER TRAVERSAL

4.PREORDER TRAVERSAL

5.POSTORDER TRAVERSAL

6.EXIT

ENTER YOUR CHOICE:2

The elements are:1 2 3 9 11 13 17 25 57 90

Do you want to continue(1/0):1

1.INSERT

2.DISPLAY

3.INORDER TRAVERSAL

4.PREORDER TRAVERSAL

5.POSTORDER TRAVERSAL

6.EXIT

ENTER YOUR CHOICE:3

After inorder traversal

1 2 3 9 11 13 17 25 57 90

Do you want to continue(1/0):1

1.INSERT

2.DISPLAY

3.INORDER TRAVERSAL

4.PREORDER TRAVERSAL

5.POSTORDER TRAVERSAL

6.EXIT

ENTER YOUR CHOICE:4

After preorder traversal

11 2 1 9 3 13 57 25 17 90

Do you want to continue(1/0):1

1.INSERT

2.DISPLAY

3.INORDER TRAVERSAL

4.PREORDER TRAVERSAL

5.POSTORDER TRAVERSAL

6.EXIT

ENTER YOUR CHOICE:5

After postorder traversal

1 3 9 2 17 25 90 57 13 11

Do you want to continue(1/0):0

**PROGRAM NO: 13**

**AIM: Write a program to implement binary search tree operations.**

**PROGRAM:**

#include<iostream>

using namespace std;

#define TRUE 1

#define FALSE 0

class btree1

{

private:

struct btreenode

{

btreenode \*leftchild;

int data;

btreenode \*rightchild;

}\*root;

public:

btree1();

void buildtree(int num);

static void insert(btreenode \*\*sr,int);

static void search(btreenode \*\*sr,int num,btreenode \*\*par,btreenode \*\*x,int \*found);

void remove(int num);

static void inorder(btreenode \*sr);

void display();

static void rem(btreenode \*\*sr,int num);

~btree1();

static void del(btreenode \*sr);

};

btree1::btree1()

{

root=NULL;

}

void btree1::buildtree(int num)

{

insert(&root,num);

}

void btree1::insert(btreenode \*\*sr,int num)

{

if(\*sr==NULL)

{

\*sr=new btreenode;

(\*sr)->leftchild=NULL;

(\*sr)->data=num;

(\*sr)->rightchild=NULL;

}

else

{

if(num<(\*sr)->data)

insert(&((\*sr)->leftchild),num);

else

insert(&((\*sr)->rightchild),num);

}

}

void btree1::remove(int num)

{

rem(&root,num);

}

void btree1::rem(btreenode \*\*sr,int num)

{

int found;

btreenode \*parent,\*x,\*xsucc;

if(\*sr==NULL)

{

cout<<"\n Tree is empty\n";

return;

}

parent=x=NULL;

search(sr,num,&parent,&x,&found);

if(found==FALSE)

{

cout<<"\n Data to be deleted is not present\n";

return;

}

if(x->leftchild!=NULL&&x->rightchild!=NULL)

{

parent=x;

xsucc=x->rightchild;

while(xsucc->leftchild!=NULL)

{

parent=xsucc;

xsucc=xsucc->leftchild;

}

x->data=xsucc->data;

x=xsucc;

}

if(x->leftchild==NULL&&x->rightchild==NULL)

{

if(parent->rightchild==x)

parent->rightchild=NULL;

else

parent->leftchild=NULL;

delete x;

return;

}

if(x->leftchild==NULL&&x->rightchild!=NULL)

{

if(parent->leftchild==x)

parent->leftchild=x->rightchild;

else

parent->rightchild=x->rightchild;

delete x;

return;

}

if(x->leftchild!=NULL&&x->rightchild==NULL)

{

if(parent->leftchild==x)

parent->leftchild=x->rightchild;

else

parent->rightchild=x->leftchild;

delete x;

return;

}

}

void btree1::search(btreenode \*\*sr,int num,btreenode \*\*par,btreenode \*\*x,int \*found)

{

btreenode \*q;

q=\*sr;

\*found=FALSE;

\*par=NULL;

while(q!=NULL)

{

if(q->data==num)

{

\*found=TRUE;

\*x=q;

return;

}

\*par=q;

if(q->data>num)

q=q->leftchild;

else

q=q->rightchild;

}

}

void btree1::display()

{

inorder(root);

}

void btree1::inorder(btreenode \*sr)

{

if(sr!=NULL)

{

inorder(sr->leftchild);

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

inorder(sr->rightchild);

}

}

btree1::~btree1()

{

del(root);

}

void btree1::del(btreenode \*sr)

{

if(sr!=NULL)

{

del(sr->leftchild);

del(sr->rightchild);

}

delete sr;

}

int main()

{

cout<<"\n BINARY SEARCH TREE\n----------------------";

cout<<"\n";

btree1 bt;

int req,i=0,num;

int a[]={17,52,13,24,55,66,78,80,91};

while(i<=8)

{

bt.buildtree(a[i]);

i++;

}

cout<<"\nBinary tree before deletion:\n";

bt.display();

bt.remove(17);

cout<<"\n Binary tree after deletion:\n";

bt.display();

bt.remove(55);

cout<<"\n Binary tree after deletion:\n";

bt.display();

bt.remove(80);

cout<<"\n Binary tree after deletion:\n";

bt.display();

}

**OUTPUT:**

BINARY SEARCH TREE

----------------------

Binary tree before deletion:

13 17 24 52 55 66 78 80 91

Binary tree after deletion:

13 24 52 55 66 78 80 91

Binary tree after deletion:

13 24 52 66 78 80 91

Binary tree after deletion:

13 24 52 66 78 91

**PROGRAM NO: 14**

**AIM: Write a program to sort array elements using heap sort.**

**PROGRAM:**

#include<iostream>

using namespace std;

const int MAX=10;

class array

{

private:

int arr[MAX];

int count;

public:

array();

void add(int num);

void makeheap();

void heapsort();

void display();

};

array::array()

{

count=0;

int i;

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

arr[MAX]=0;

}

void array::add(int num)

{

if(count<MAX)

{

arr[count]=num;

count ++;

}

else

cout<<"Array is full";

}

void array::makeheap()

{

for(int i=1;i<count;i++)

{

int val=arr[i];

int s=i;

int f=(s-1)/2;

while(s>0&&arr[f]<val)

{

arr[s]=arr[f];

s=f;

f=(s-1)/2;

}

arr[s]=val;

}

}

void array::heapsort()

{

for(int i=count-1;i>0;i--)

{

int ivalue=arr[i];

arr[i]=arr[0];

int f=0;

int s;

if(i==1)

s=-1;

else

s=1;

if(i>2&&arr[2]>arr[1])

s=2;

while(s>=0&& ivalue<arr[s])

{

arr[f]=arr[s];

f=s;

s=2\*f+1;

if(s+1<=i-1&&arr[s]<arr[s+1])

s++;

if(s>i-1)

s=-1;

}

arr[f]=ivalue;

}

}

void array::display()

{

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

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

cout<<"\n";

}

main()

{

array a;

int n,i,num;

cout<<"\*\*\*\*\*\*\*\*\*HEAP SORT\*\*\*\*\*\*\*\*\*\n";

cout<<"Enter the size of the array:";

cin>>n;

cout<<"Elements are:";

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

{

cin>>num;

a.add(num);

}

a.makeheap();

cout<<"Before sorting:\n";

a.display();

cout<<"\n";

a.heapsort();

cout<<"After sorting:\n";

a.display();

}

**OUTPUT:**

\*\*\*\*\*\*\*\*\*HEAP SORT\*\*\*\*\*\*\*\*\*

Enter the size of the array:5

Elements are:78 23 45 10 90

Before sorting:

90 78 45 10 23

After sorting:

10 23 45 78 90

**PROGRAM NO: 15**

**AIM: Write a program to implement dijkstra’s algorithm.**

**PROGRAM:**

#include<iostream>

using namespace std;

const int INF=9999;

int main()

{

cout<<"\*\*\*\*\*\*\*\*DIJKSTRAS ALGORITHM\*\*\*\*\*\*\*\*\*\*";

cout<<"\n";

int a[4][4];

int cost[4][4]={7,5,0,0,7,0,0,2,0,3,0,0,4,0,1,0};

int i,j,k,n=4;

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

{

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

{

if(cost[i][j]==0)

a[i][j]=INF;

else

a[i][j]=cost[i][j];

}

}

cout<<"Adjacency matrix of cost of edges\n";

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

{

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

{

cout<<a[i][j]<<"\t";

}

cout<<"\n";

}

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

{

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

{

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

{

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

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

}

}

}

cout<<"\n";

cout<<"Adjacency matrix of lowest cost between the vertices:\n";

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

{

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

{

cout<<a[i][j]<<"\t";

}

cout<<"\n";

}

}

**OUTPUT:**

\*\*\*\*\*\*\*\*DIJKSTRAS ALGORITHM\*\*\*\*\*\*\*\*\*\*

Adjacency matrix of cost of edges

7 5 9999 9999

7 9999 9999 2

9999 3 9999 9999

4 9999 1 9999

Adjacency matrix of lowest cost between the vertices:

7 5 8 7

6 6 3 2

9 3 6 5

4 4 1 6

**PROGRAM NO: 16**

**AIM: Write a program to sort array elements using bubble sort.**

**PROGRAM:**

#include<iostream>

using namespace std;

class bubble

{

int i,a[10],j,n;

public:

void getdata();

void sort();

void display();

};

void bubble::getdata()

{

cout<<"Enter limit:";

cin>>n;

cout<<"Enter the elements:";

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

{

cin>>a[i];

}

}

void bubble::sort()

{

int temp;

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;

}

}

}

}

void bubble::display()

{

cout<<"Sorted array is:";

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

{

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

}

}

main()

{

bubble ob;

ob.getdata();

ob.sort();

ob.display();

}

**OUTPUT:**

Enter limit:7

Enter the elements:67 34 12 0 7 2 4

Sorted array is:0 2 4 7 12 34 67

**PROGRAM NO: 17**

**AIM: Write a program to search a particular element in a array.**

**PROGRAM:**

#include<iostream>

using namespace std;

class search

{

int n,i,j,a[10],k;

public:

void getdata();

void sort();

void binary();

void display();

};

void search::getdata()

{

cout<<"Enter limit:";

cin>>n;

cout<<"Enter the elements:";

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

{

cin>>a[i];

}

}

void search::sort()

{

int temp;

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;

}

}

}

}

void search::binary()

{

int mid,k,lower=0,upper=n-1,f=0;

cout<<"Enter the element to search:";

cin>>k;

while(lower<=upper)

{

mid=(lower+upper)/2;

if(a[mid]==k)

{

f=1;

break;

}

if(a[mid]<k)

{

lower=mid+1;

}

else if(a[mid]>k)

{

upper=mid-1;

}

}

if(f==1)

cout<<"Element "<<k<<" is found in position "<<mid+1<<"\n";

else

cout<<"Element not found\n";

}

void search::display()

{

cout<<"Sorted array is:";

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

{

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

}

cout<<"\n";

}

main()

{

search ob;

ob.getdata();

ob.sort();

ob.display();

ob.binary();

}

**OUTPUT:**

Enter limit:7

Enter the elements:89 45 32 12 10 43 66

Sorted array is:10 12 32 43 45 66 89

Enter the element to search:66

Element 66 is found in position 6

Enter limit:5

Enter the elements:10 40 15 67 77

Sorted array is:10 15 40 67 77

Enter the element to search:100

Element not found

**PROGRAM NO: 18**

**AIM: Write a program to sort array elements using quick sort.**

**PROGRAM:**

#include<iostream>

using namespace std;

const int MAX=10;

class quick

{

private:

int a[MAX];

int count;

public:

quick();

void add(int item);

int getcount();

static int split(int \*,int,int);

void quicksort(int lower,int upper);

void display();

};

quick::quick()

{

count=0;

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

a[i]=0;

}

void quick::add(int item)

{

if(count<MAX)

{

a[count]=item;

count++;

}

else

cout<<"Quick is full";

}

int quick::getcount()

{

return count;

}

void quick::quicksort(int lower,int upper)

{

if(upper>lower)

{

int i=split(a,lower,upper);

quicksort(lower,i-1);

quicksort(i+1,upper);

}

}

int quick::split(int \*a,int lower,int upper)

{

int i,p,q,t;

p=lower+1;

q=upper;

i=a[lower];

while(q>=p)

{

while(a[p]<i)

p++;

while(a[q]>i)

q--;

if(q>p)

{

t=a[p];

a[p]=a[q];

a[q]=t;

}

}

t=a[lower];

a[lower]=a[q];

a[q]=t;

return q;

}

void quick::display()

{

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

{

cout<<a[i];

cout<<"\n";

}

}

main()

{

quick ob;

int s,n;

cout<<"Enter the size:";

cin>>s;

cout<<"Enter the elements:";

for(int i=1;i<=s;i++)

{

cin>>n;

ob.add(n);

}

cout<<"\*\*\*Quick sort\*\*\*\*\n";

cout<<"Quick before sort:\n;

ob.display();

int c=ob.getcount();

ob.quicksort(0,c-1);

cout<<"Quick after quick sorting:\n";

ob.display();

}

**OUTPUT:**

Enter the size:7

Enter the elements:34 56 12 90 1 23 11

\*\*\*Quick sort\*\*\*\*

Quick before sort:

34

56

12

90

1

23

11

Quick after quick sorting:

1

11

12

23

34

56

90

**PROGRAM NO: 19**

**AIM: Write a program to sort array elements using merge sort.**

**PROGRAM:**

#include<iostream>

using namespace std;

class array

{

private:

int \*arr;

int size;

int count;

public:

array();

array(int sz);

void add(int num);

static void sort(int \*a,int sz);

void merge(array &a,array &b);

void display();

~array();

};

array::array()

{

count=size=0;

arr=NULL;

}

array::array(int sz)

{

count=0;

size=sz;

arr=new int[sz];

}

void array::add(int num)

{

if(count<size)

{

arr[count]=num;

count++;

}

else

cout<<"ARRAY FULL\n";

}

void array::display()

{

cout<<"Elements are:";

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

cout<<arr[j]<<" ";

cout<<"\n";

}

void array::merge(array &a,array &b)

{

sort(a.arr,a.size);

sort(b.arr,b.size);

size=a.count+b.count;

arr=new int[size];

int i,j,k;

for(i=j=k=0;j<a.count||k<b.count;)

{

if(a.arr[j]<=b.arr[k])

arr[i++]=a.arr[j++];

else

arr[i++]=b.arr[k++];

count++;

if(j==a.count||k==b.count)

break;

}

for(;k<b.count;)

{

arr[i++]=b.arr[k++];

count++;

}

}

void array::sort(int \*a,int sz)

{

int temp;

for(int i=0;i<=sz-2;i++)

{

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

{

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

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

}

array::~array()

{

delete arr;

}

main()

{

cout<<"\*\*\*\*\* MERGE SORT\*\*\*\*\*\*\*\*\*\n";

int i,n1,n2,num1,num2;

cout<<"Enter the limit for first array:";

cin>>n1;

cout<<"\n";

array a(n1);

cout<<"Enter number:";

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

{

cin>>num1;

a.add(num1);

}

cout<<"\n";

cout<<" First array:"<<endl;

a.display();

cout<<"\n";

cout<<"Enter the limit for second array:";

cin>>n2;

cout<<"\n";

array b(n2);

cout<<"Enter numbers:";

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

{

cin>>num2;

b.add(num2);

}

cout<<"\n";

cout<<"Second array:";

b.display();

cout<<"\n";

array c;

c.merge(a,b);

cout<<"Array after sorting:\n";

c.display();

cout<<"\n";

}

**OUTPUT:**

\*\*\*\*\* MERGE SORT\*\*\*\*\*\*\*\*\*

Enter the limit for first array:6

Enter number:12 90 87 1 2 67

First array:

Elements are:12 90 87 1 2 67

Enter the limit for second array:7

Enter numbers:12 123 78 56 45 3 9

Second array:Elements are:12 123 78 56 45 3 9

Array after sorting:

Elements are:1 2 3 9 12 12 45 56 67 78 87 90 123

**PROGRAM NO: 20**

**AIM: Write a program to implement depth first search algorithm in graph.**

**PROGRAM:**

#include<iostream>

using namespace std;

#define TRUE 1

#define FALSE 0

const int MAX=8;

struct node

{

int data;

node \*next;

};

class graph

{

private:

int visited[MAX];

public:

graph();

void dfs(int v,node\*\* p);

node\* getnode\_write(int val);

void del(node \*n);

};

graph::graph()

{

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

visited[i]=FALSE;

}

void graph::dfs(int v,node \*\*p)

{

node \*t;

visited[v-1]=TRUE;

cout<<v<<"\t";

t=\*(p+v-1);

while(t!=NULL)

{

if(visited[t->data-1]==FALSE)

dfs(t->data,p);

else

t=t->next;

}

}

node\* graph::getnode\_write(int val)

{

node \*newnode=new node;

newnode->data=val;

return newnode;

}

void graph::del(node \*n)

{

node \*temp;

while(n!=NULL)

{

temp=n->next;

delete n;

n=temp;

}

}

int main()

{

cout<<"\*\*\*\*\*\*DEPTH FIRST SEARCH\*\*\*\*\*\*\*\*\*\*\n";

node \*arr[MAX];

node \*v1,\*v2,\*v3,\*v4;

graph g;

v1=g.getnode\_write(2);

arr[0]=v1;

v1->next=v2=g.getnode\_write(8);

v2->next=NULL;

v1=g.getnode\_write(1);

arr[1]=v1;

v1->next=v2=g.getnode\_write(3);

v2->next=v3=g.getnode\_write(8);

v3->next=NULL;

v1=g.getnode\_write(2);

arr[2]=v1;

v1->next=v2=g.getnode\_write(4);

v2->next=v3=g.getnode\_write(8);

v3->next=NULL;

v1=g.getnode\_write(3);

arr[3]=v1;

v1->next=v2=g.getnode\_write(7);

v2->next=NULL;

v1=g.getnode\_write(6);

arr[4]=v1;

v1->next=v2=g.getnode\_write(7);

v2->next=NULL;

v1=g.getnode\_write(5);

arr[5]=v1;

v2->next=NULL;

v1=g.getnode\_write(4);

arr[6]=v1;

v1->next=v2=g.getnode\_write(5);

v2->next=v3=g.getnode\_write(8);

v3->next=NULL;

v1=g.getnode\_write(1);

arr[7]=v1;

v1->next=v2=g.getnode\_write(2);

v2->next=v3=g.getnode\_write(3);

v3->next=v4=g.getnode\_write(7);

v4->next=NULL;

cout<<endl;

g.dfs(1,arr);

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

g.del(arr[i]);

}

**OUTPUT:**

\*\*\*\*\*\*DEPTH FIRST SEARCH\*\*\*\*\*\*\*\*\*\*

1 2 3 4 7 5 6 8

**COMPUTER GRAPHICS**

**PROGRAM NO: 1**

**AIM: Program to draw points on screen.**

**PROGRAM:**

#include<GL/glut.h>

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(10.0);

glBegin(GL\_POINTS);

glVertex2i(50,50);

glColor3f(1.0,0.0,0.0);

glVertex2i(20,20);

glColor3f(0.0,0.0,0.0);

glEnd();

glFlush();

}

main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Point Plotting");

glClearColor(0.5,0.0,0.5,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM NO: 2**

**AIM**: **Program to draw line on screen.**

**PROGRAM:**

#include<GL/glut.h>

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(10.0);

glBegin(GL\_LINES);

glColor3f(0.0,1.0,0.0);

glVertex2i(60,60);

glVertex2i(20,20);

glColor3f(0.0,0.0,1.0);

glEnd();

glFlush();

}

main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Line Plotting");

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

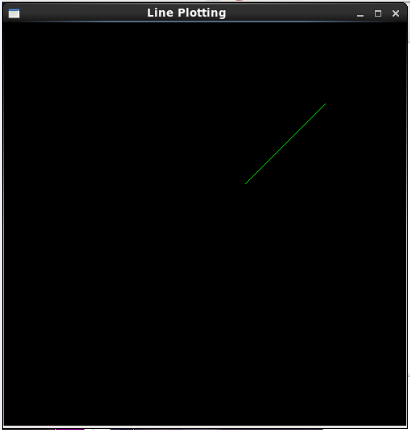
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM NO: 3**

**AIM: Program to draw a line using DDA line drawing algorithm.**

**PROGRAM:**

#include<stdio.h>

#include<GL/glut.h>

float x,y,x1,z1,x2,y2,dx,dy,step;

void dda()

{

int xinc,yinc,k;

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(9.0);

glColor3f(1.0,0.0,0.0);

dx=x2-x1;

dy=y2-z1;

if(abs(dx)>abs(dy))

step=abs(dx);

else

step=abs(dy);

xinc=dx/(float)step;

yinc=dy/(float)step;

x=x1;

y=z1;

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

{

glBegin(GL\_LINES);

glVertex2f(x,y);

x=x+xinc;

y=y+yinc;

glVertex2f(x,y);

}

glEnd();

glFlush();

}

main(int argc,char \*\*argv)

{

printf("\nEnter the cordinates of x1 and z1:");

scanf("%f%f",&x1,&z1);

printf("\nEnter the cordinates of x2 and y2:");

scanf("%f%f",&x2,&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Line\_Drawing");

glClearColor(0.0,1.0,1.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(dda);

glutMainLoop();

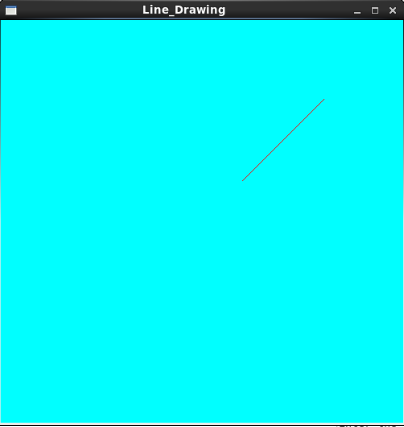
return 0;

}

**OUTPUT:**

Enter the coordinates of x1 and z1:20 20

Enter the coordinates of x2 and y2:60 60



**PROGRAM NO: 4**

**AIM: Program to draw line using bresenham’s line drawing algorithm**

**PROGRAM:**

#include<stdio.h>

#include<GL/glut.h>

float x1,x2,z1,y2,x,y,step,p,dx,dy;

void bresenham()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(5.0);

glColor3f(1.0,0.0,0.0);

int k;

dx=x2-x1;

dy=y2-z1;

step=dx-1;

p=2\*(dy-dx);

x=x1;

y=z1;

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

{

glBegin(GL\_POINTS);

glVertex2f(x,y);

if(p<0)

{

x=x+1;

p=p+(2\*dy);

}

else

{

x=x+1;

y=y+1;

p=p+2\*(dy-dx);

}

glVertex2f(x,y);

}

glEnd();

glFlush();

}

main(int argc,char \*\*argv)

{

printf("\nEnter the cordinates of x1 and z1:");

scanf("%f%f",&x1,&z1);

printf("\nEnter the cordinates of x2 and y2:");

scanf("%f%f",&x2,&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Line\_Drawing");

glClearColor(0.0,1.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(bresenham);

glutMainLoop();

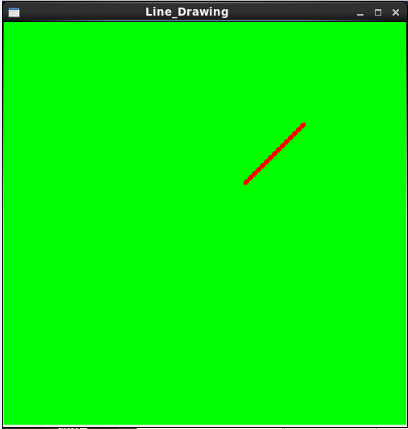
return 0;

}

**OUTPUT:**

Enter the coordinates of x1 and z1:20 20

Enter the coordinates of x2 and y2:50 50



**PROGRAM NO: 5**

**AIM: Program to fill a region using flood fill algorithm.**

**PROGRAM:**

#include<GL/glut.h>

void floodfill(float,float,float[],float[]);

float fill[3]={1.0,0.0,1.0},old[3]={0.0,0.0,1.0};

void display()

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3fv(old);

glBegin(GL\_POLYGON);

glVertex2i(100,150);

glVertex2i(400,150);

glVertex2i(400,350);

glVertex2i(100,350);

glEnd();

glFlush();

floodfill(200.0,160.0,fill,old);

glFlush();

}

void floodfill(float x,float y,float fill[3],float old[3])

{

float pix[3];

glReadPixels(x,y,1.0,1.0,GL\_RGB,GL\_FLOAT,pix);

if(pix[0]==old[0]&&pix[1]==old[1]&&pix[2]==old[2])

{

glBegin(GL\_POINTS);

glColor3fv(fill);

glVertex2f(x,y);

glEnd();

glFlush();

floodfill(x-1,y,fill,old);

floodfill(x+1,y,fill,old);

floodfill(x,y+1,fill,old);

floodfill(x,y-1,fill,old);

}

}

int main(int argc,char \*argv[])

{

glutInit(&argc,argv);

glutInitWindowSize(640,480);

glutCreateWindow("Flood Fill");

glutDisplayFunc(display);

glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM NO: 6**

**AIM: Program to draw a circle.**

**PROGRAM:**

#include<stdio.h>

#include<GL/glut.h>

void circle\_plotting(float xx,float yy);

float r,xc,yc,x,y,p;

void circle\_algorithm()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,1.0);

glPointSize(4.0);

x=0;

y=r;

p=1-r;

glBegin(GL\_POINTS);

circle\_plotting(x,y);

while(x<=y)

{

if(p<0)

{

x=x+1;

p=p+(2\*x)+1;

}

else

{

x=x+1;

y=y-1;

p=p+(2\*(x-y))+1;

}

circle\_plotting(x,y);

}

glEnd();

glFlush();

}

void circle\_plotting(float xx,float yy)

{

glVertex2f(xc+xx,yc+yy);

glVertex2f(xc-xx,yc-yy);

glVertex2f(xc-xx,yc+yy);

glVertex2f(xc+xx,yc-yy);

glVertex2f(xc+yy,yc+xx);

glVertex2f(xc-yy,yc+xx);

glVertex2f(xc+yy,yc-xx);

glVertex2f(xc-yy,yc-xx);

}

main(int argc,char \*\*argv)

{

printf("Enter radious :");

scanf("%f",&r);

printf("Enter center point xc and yc: ");

scanf("%f%f",&xc,&yc);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow(" circle\_Drawing");

glClearColor(1.0,1.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(circle\_algorithm);

glutMainLoop();

return 0;

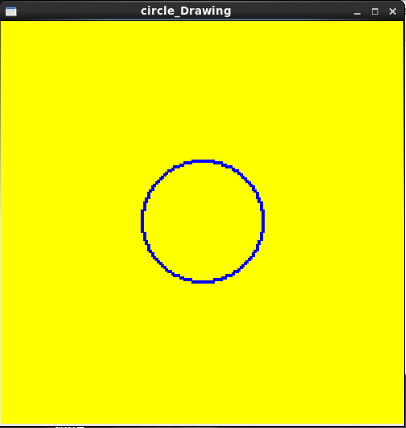
}

**OUTPUT:**

Enter radious :30

Enter center point xc and yc: 0.4

0.5



**PROGRAM NO: 7**

**AIM: Program to implement line clipping using cohen-sutherland algorithm.**

**PROGRAM:**

#include<stdio.h>

#include<GL/glut.h>

#define outcode int

double xmin=50,ymin=50,xmax=100,ymax=100;

double xvmin=200,yvmin=200,xvmax=300,yvmax=300;

const int RIGHT=8;

const int LEFT=2;

const int TOP=4;

const int BOTTOM=1;

outcode computeoutcode(double x,double y);

void cohen\_clipping(double x0,double y0,double x1,double y1)

{

outcode outcode0,outcode1,outcodeout;

int accept=0,done=0;

outcode0=computeoutcode(x0,y0);

outcode1=computeoutcode(x1,y1);

do

{

if(!(outcode0|outcode1))

{

accept=1;

done=1;

}

else if(outcode0 &outcode1)

done=1;

else

{

double x,y;

outcodeout=outcode0?outcode0:outcode1;

if(outcodeout & TOP)

{

x=x0+(x1-x0)\*(ymax-y0)/(y1-y0);

y=ymax;

}

else if(outcodeout & BOTTOM)

{

x=x0+(x1-x0)\*(ymin-y0)/(y1-y0);

y=ymin;

}

else if(outcodeout & RIGHT)

{

y=y0+(y1-y0)\*(xmax-x0)/(x1-x0);

x=xmax;

}

else

{

y=y0+(y1-y0)+(xmin-x0)/(x1-x0);

x=xmin;

}

if(outcodeout==outcode0)

{

x0=x;

y0=y;

outcode0=computeoutcode(x0,y0);

}

else

{

x1=x;

y1=y;

outcode1=computeoutcode(x1,y1);

}

}

}while(!done);

if(accept)

{

double sx=(xvmax-xvmin)/(xmax-xmin);

double sy=(yvmax-yvmin)/(ymax-ymin);

double vx0=xvmin+(x0-xmin)\*sx;

double vy0=yvmin+(y0-ymin)\*sy;

double vx1=xvmin+(x1-xmin)\*sx;

double vy1=yvmin+(y1-ymin)\*sy;

glColor3f(0.0,1.0,0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xvmin,yvmin);

glVertex2f(xvmax,yvmin);

glVertex2f(xvmax,yvmax);

glVertex2f(xvmin,yvmax);

glEnd();

glColor3f(0.0,0.0,1.0);

glBegin(GL\_LINES);

glVertex2d(vx0,vy0);

glVertex2d(vx1,vy1);

glEnd();

}

}

outcode computeoutcode(double x,double y)

{

outcode code=0;

if(y>ymax)

code |=TOP;

if(y<ymin)

code |=BOTTOM;

if(x>xmax)

code |=RIGHT;

if(x<xmin)

code |=TOP;

return code;

}

void display()

{

double x0=120,y0=10,x1=40,y1=130;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2d(x0,y0);

glVertex2d(x1,y1);

glVertex2d(60,20);

glVertex2d(80,120);

glEnd();

glColor3f(1.0,0.0,1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xmin,ymin);

glVertex2f(xmax,ymin);

glVertex2f(xmax,ymax);

glVertex2f(xmin,ymax);

glEnd();

cohen\_clipping(x0,y0,x1,y1);

cohen\_clipping(60,20,80,120);

glFlush();

}

void init()

{

glClearColor(0.0,0.0,0.0,1.0);

glColor3f(1.0,0.0,0.0);

glPointSize(1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);

}

int main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutCreateWindow("Cohen Sutherland line clipping");

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutDisplayFunc(display);

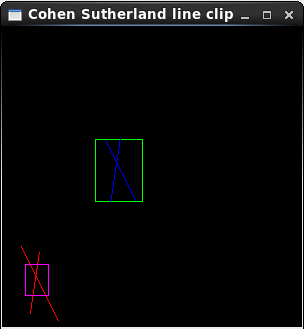
init();

glutMainLoop();

return 0;

}

**OUTPUT:**

****

**PROGRAM NO: 8**

**AIM: Program to implement polygon clipping.**

**PROGRAM:**

#include <GL/glut.h>

struct Point

{

float x,y;

} w[4],oVer[4];

int Nout;

void drawPoly(Point p[],int n)

{

glBegin(GL\_POLYGON);

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

glVertex2f(p[i].x,p[i].y);

glEnd();

}

bool insideVer(Point p)

{

if((p.x>=w[0].x)&&(p.x<=w[2].x))

if((p.y>=w[0].y)&&(p.y<=w[2].y))

return true;

return false;

}

void addVer(Point p)

{

oVer[Nout]=p;

Nout=Nout+1;

}

Point getInterSect(Point s,Point p,int edge)

{

Point in;

float m;

if(w[edge].x==w[(edge+1)%4].x)

{ //Vertical Line

m=(p.y-s.y)/(p.x-s.x);

in.x=w[edge].x;

in.y=in.x\*m+s.y;

}

else

{

m=(p.y-s.y)/(p.x-s.x);

in.y=w[edge].y;

in.x=(in.y-s.y)/m;

}

return in;

}

void clipAndDraw(Point inVer[],int Nin)

{

Point s,p,interSec;

for(int i=0;i<4;i++)

{

Nout=0;

s=inVer[Nin-1];

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

{

p=inVer[j];

if(insideVer(p)==true)

{

if(insideVer(s)==true)

{

addVer(p);

}

else

{

interSec=getInterSect(s,p,i);

addVer(interSec);

addVer(p);

}

}

else

{

if(insideVer(s)==true)

{

interSec=getInterSect(s,p,i);

addVer(interSec);

}

}

s=p;

}

inVer=oVer;

Nin=Nout;

}

drawPoly(oVer,4);

}

void init()

{

glClearColor(0.0f,1.0f,1.0f,0.0f);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0,100.0,0.0,100.0,0.0,100.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

w[0].x =15,w[0].y=10;

w[1].x =15,w[1].y=40;

w[2].x =40,w[2].y=40;

w[3].x =40,w[3].y=10;

}

void display(void)

{

Point inVer[4];

init();

glColor3f(0.0f,1.0f,0.0f);

drawPoly(w,4);

glColor3f(0.0f,1.0f,0.0f);

inVer[0].x =10,inVer[0].y=40;

inVer[1].x =10,inVer[1].y=30;

inVer[2].x =30,inVer[2].y=30;

inVer[3].x =30,inVer[3].y=40;

drawPoly(inVer,4);

glColor3f(0.0f,0.0f,1.0f);

clipAndDraw(inVer,4);

glFlush();

}

int main(int argc,char \*argv[])

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(400,400);

glutInitWindowPosition(100,100);

glutCreateWindow("Polygon Clipping!");

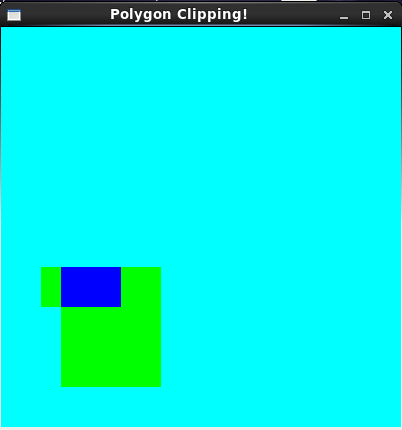
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM NO: 9**

**AIM: Two dimensional transformations.**

**PROGRAM:**

#include<stdio.h>

#include<math.h>

#include<GL/glut.h>

int ch;

float x1=0.5,x2=0.8,x3=0.8,x4=0.5,y=0.5,y2=0.5,y3=0.8,y4=0.8;

float X1,X2,X3,X4,Y,Y2,Y3,Y4;

void display(void)

{

float tx,ty;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.3,1.3,0.3);

glPointSize(10.0);

glBegin(GL\_POLYGON);

glVertex2f(x1,y);

glVertex2f(x2,y2);

glVertex2f(x3,y3);

glVertex2f(x4,y4);

glEnd();

glColor3f(0.8,0.0,0.0);

glBegin(GL\_POLYGON);

glVertex2f(X1,Y);

glVertex2f(X2,Y2);

glVertex2f(X3,Y3);

glVertex2f(X4,Y4);

glEnd();

glFlush();

}

void translate()

{

float tx,ty;

printf("ENTER tx AND ty VALUE\n");

scanf("%f%f",&tx,&ty);

X1=x1+tx;

X2=x2+tx;

X3=x3+tx;

Y=y+ty;

Y2=y2+ty;

Y3=y3+ty;

X4=x4+tx;

Y4=y4+ty;

}

void rotate()

{

int theta;

printf("ENTER AN ANGLE\n");

scanf("%d",&theta);

X1=x1\*cos(theta)-y\*sin(theta);

X2=x2\*cos(theta)-y2\*sin(theta);

X3=x3\*cos(theta)-y3\*sin(theta);

X4=x4\*cos(theta)-y4\*sin(theta);

Y=x1\*sin(theta)+y\*cos(theta);

Y2=x2\*sin(theta)+y2\*cos(theta);

Y3=x3\*sin(theta)+y3\*cos(theta);

Y4=x4\*sin(theta)+y4\*cos(theta);

}

void scale()

{

float sx,sy;

printf("ENTER sx AND sy VALUE\n");

scanf("%f%f",&sx,&sy);

X1=x1\*sx;

X2=x2\*sx;

X3=x3\*sx;

X4=x4\*sx;

Y=y\*sy;

Y2=y2\*sy;

Y3=y3\*sy;

Y4=y4\*sy;

}

main(int argc,char \*\*argv)

{

printf("2D TRANSFORMATION OPERATIONS\n");

printf("1:TRANSLATION\n");

printf("2:ROTATION\n");

printf("3:SCALING\n");

printf("ENTER UR CHOICE\n");

scanf("%d",&ch);

switch(ch)

{

case 1: translate();

break;

case 2: rotate();

break;

case 3: scale();

break;

}

glutInit(&argc,argv);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("2D TRANSFORMATION");

glClearColor(0.0,1.0,1.3,0.0);

glutDisplayFunc(display);

glutMainLoop();

}

**OUTPUT:**

2D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

3:SCALING

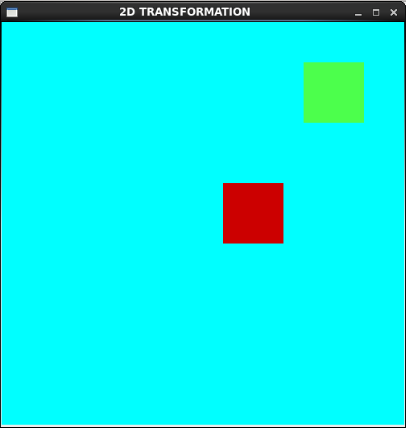
ENTER UR CHOICE

1

ENTER tx AND ty VALUE

-0.4

-0.6



2D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

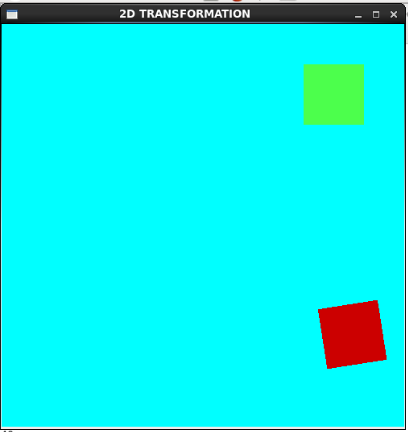
3:SCALING

ENTER UR CHOICE

2

ENTER AN ANGLE

30



2D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

3:SCALING

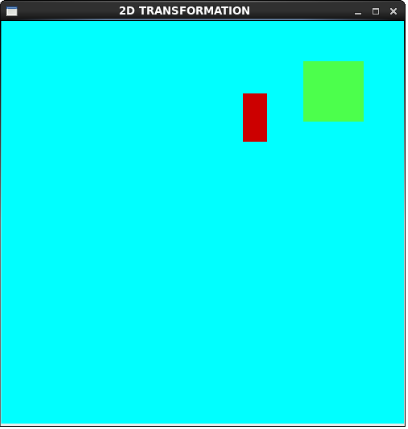
ENTER UR CHOICE

3

ENTER sx AND sy VALUE

0.4

0.8



**PROGRAM NO: 10**

**AIM: Three dimensional viewing.**

**PROGRAM:**

#include<GL/glut.h>

GLint winWidth=600,winHeight=600;

GLfloat x0=100.0,y0=50.0,z0=50.0;

GLfloat xref=50.0,yref=50.0,zref=0.0;

GLfloat Vx=0.0,Vy=1.0,Vz=0.0;

GLfloat xwMin=-40,ywMin=-60,xwMax=40.0,ywMax=60.0;

GLfloat dnear=25.0,dfar=125.0;

void init(void)

{

glClearColor(0.0,0.0,0.0,0.0);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(x0,y0,z0,xref,yref,zref,Vx,Vy,Vz);

glMatrixMode(GL\_PROJECTION);

glFrustum(xwMin,xwMax,ywMin,ywMax,dnear,dfar);

}

void displayFun(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,1.0);

glPolygonMode(GL\_FRONT,GL\_FILL);

glPolygonMode(GL\_BACK,GL\_LINE);

glBegin(GL\_QUADS);

glVertex3f(0.0,0.0,0.0);

glVertex3f(100.0,0.0,0.0);

glVertex3f(100.0,100.0,0.0);

glVertex3f(0.0,100.0,0.0);

glEnd();

glFlush();

}

void reshapeFun(GLint newWidth,GLint newHeight)

{

glViewport(0,0,newWidth,newHeight);

winWidth=newWidth;

winHeight=newHeight;

}

int main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowPosition(50,50);

glutInitWindowSize(winWidth,winHeight);

glutCreateWindow("Perspective view of a square");

init();

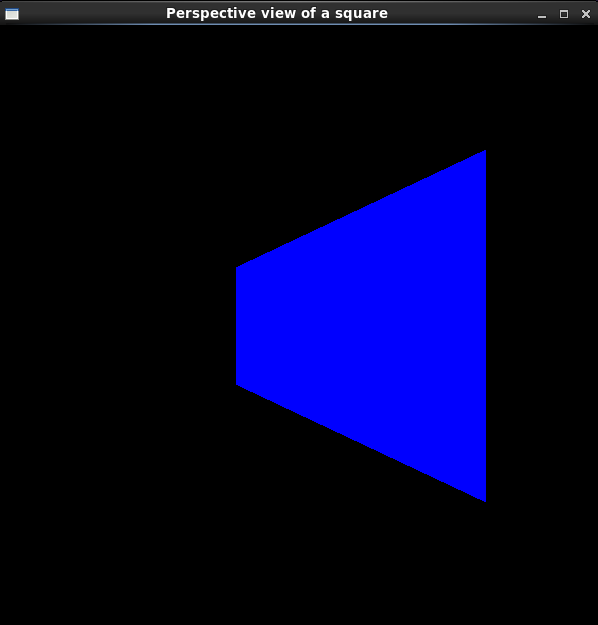
glutDisplayFunc(displayFun);

glutReshapeFunc(reshapeFun);

glutMainLoop();

}

**OUTPUT:**

****

**PROGRAM NO: 11**

**AIM: Three dimensional transformations.**

**PROGRAM:**

#include<stdio.h>

#include<math.h>

#include<GL/glut.h>

int ch;

float x1=0.5,x2=0.8,x3=0.8,x4=0.5,y=0.5,y2=0.5,y3=0.8,y4=0.8,z1=0.6,z2=0.4,z3=0.7,z4=0.2;

float X1,X2,X3,X4,Y,Y2,Y3,Y4,Z1,Z2,Z3,Z4;

void display()

{

float tx,ty;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.68,9.33,0.37);

glPointSize(10.0);

glBegin(GL\_POLYGON);

glVertex3f(x1,y,z1);

glVertex3f(x2,y2,z2);

glVertex3f(x3,y3,z3);

glVertex3f(x4,y4,z4);

glEnd();

glColor3f(8080,0.0,0.0);

glBegin(GL\_POLYGON);

glVertex3f(X1,Y,Z1);

glVertex3f(X2,Y2,Z2);

glVertex3f(X3,Y3,Z3);

glVertex3f(X4,Y4,Z4);

glEnd();

glFlush();

}

void translate()

{

float tx,ty,tz;

printf("ENTER tx ty AND tz VALUE\n");

scanf("%f%f%f",&tx,&ty,&tz);

X1=x1+tx;X2=x2+tx;X3=x3+tx;X4=x4+tx;

Y=y+ty;Y2=y2+ty;Y3=y3+ty;Y4=y4+ty;

Z1=z1+tz;Z2=z2+tz;Z3=z3+tz;Z4=z4+tz;

}

void rotate()

{

int theta;

printf("ENTER AN ANGLE\n");

scanf("%d",&theta);

X1=x1\*cos(theta)-y\*sin(theta);

X2=x2\*cos(theta)-y2\*sin(theta);

X3=x3\*cos(theta)-y3\*sin(theta);

X4=x4\*cos(theta)-y4\*sin(theta);

Y=x1\*sin(theta)+y\*cos(theta);

Y2=x2\*sin(theta)+y2\*cos(theta);

Y3=x3\*sin(theta)+y3\*cos(theta);

Y4=x4\*sin(theta)+y4\*cos(theta);

Z1=z1\*cos(theta)-z1\*sin(theta);

Z2=z2\*cos(theta)-z2\*sin(theta);

Z3=z3\*cos(theta)-z3\*sin(theta);

Z4=z4\*cos(theta)-z4\*sin(theta);

}

void scale()

{

float sx,sy,sz;

printf("ENTER sx ,sy AND sz VALUE\n");

scanf("%f%f%f",&sx,&sy,&sz);

X1=x1\*sx;X2=x2\*sx;X3=x3\*sx;X4=x4\*sx;

Y=y\*sy;Y2=y2\*sy;Y3=y3\*sy;Y4=y4\*sy;

Z1=z1\*sz;Z2=z2\*sz;Z3=z3\*sz;Z4=z4\*sz;

}

int main(int argc,char \*\*argv)

{

printf("3D TRANSFORMATION OPERATIONS\n");

printf("1:TRANSLATION\n");

printf("2:ROTATION\n");

printf("3:SCALING\n");

printf("ENTER UR CHOICE\n");

scanf("%d",&ch);

switch(ch)

{

case 1: translate();

break;

case 2: rotate();

break;

case 3: scale();

break;

}

glutInit(&argc,argv);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("3D TRANSFORMATION");

glClearColor(1.0,0.0,1.0,0.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**

3D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

3:SCALING

ENTER UR CHOICE

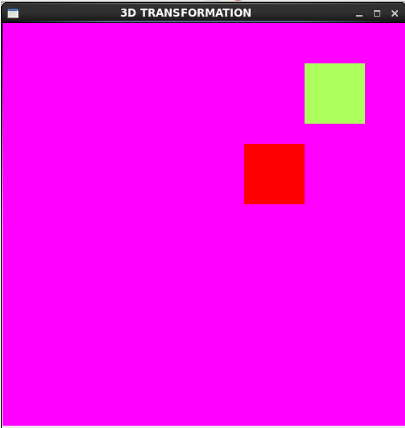
1

ENTER tx ty AND tz VALUE

-0.3

-0.4

-0.2



3D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

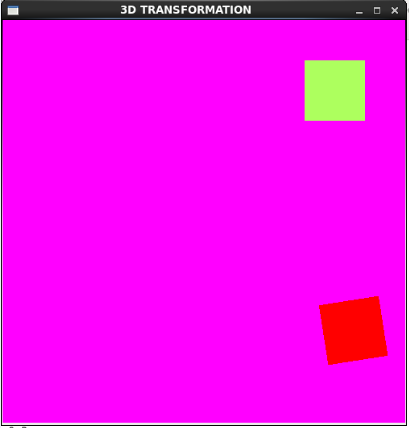
3:SCALING

ENTER UR CHOICE

2

ENTER AN ANGLE

30



3D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

3:SCALING

ENTER UR CHOICE

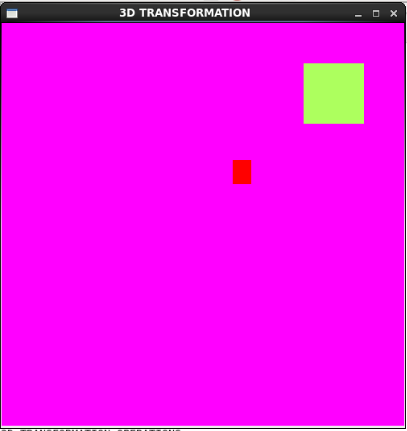
3

ENTER sx ,sy AND sz VALUE

0.3

0.4

0.3



**PROGRAM NO: 12**

**AIM: Three dimensional object representation**

**PROGRAM:**

#include<GL/glu.h>

#include<GL/glut.h>

GLfloat xRotated,yRotated,zRotated;

void displayOctahedron(void)

{

glMatrixMode(GL\_MODELVIEW);

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0,0.0,-4.5);

glColor3f(0.8,0.2,0.1);

glRotatef(xRotated,1.0,0.0,0.0);

glRotatef(yRotated,0.0,1.0,0.0);

glRotatef(zRotated,0.0,0.0,1.0);

glScalef(1.0,1.0,1.0);

glutSolidOctahedron();

glFlush();

}

void reshapeOctahedron(int x,int y)

{

if(y==0|x==0)

return;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.5);

glViewport(0,0,x,y);

}

void idleOctahedron(void)

{

yRotated+=0.03;

displayOctahedron();

}

int main(int argc,char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(400,350);

glutCreateWindow("Octahedron Rotating Animation");

glPolygonMode(GL\_FRONT\_AND\_BACK,GL\_LINE);

xRotated=yRotated=zRotated=30.0;

xRotated=33;

yRotated=40;

glClearColor(1.0,1.0,0.0,0.0);

glutDisplayFunc(displayOctahedron);

glutReshapeFunc(reshapeOctahedron);

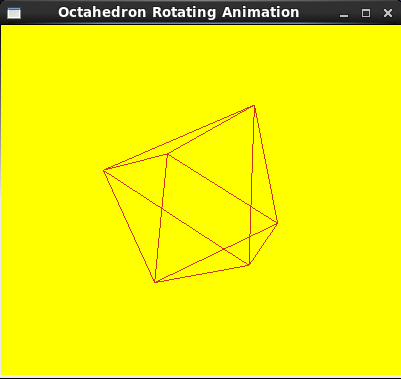
glutIdleFunc(idleOctahedron);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM NO: 13**

**AIM: Program to implement visible surface detection.**

**PROGRAM:**

#include<GL/gl.h>

#include<GL/glut.h>

void mydraw()

{

glClearColor(0.0,0.0,0.0,0.0);

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glColor3f(0.0,1.0,0.0);

glutSolidCube(7.0);

glTranslatef(0.0,0.0,-20.0);

glColor3f(0.0,0.0,1.0);

glutSolidTeapot(7.0);

glutSwapBuffers();

}

int main(int argc,char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(400,300);

glutCreateWindow("Depth Test");

glClearColor(0.0,0.0,0.0,1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0,1.333,0.01,10000.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0,0,-30);

glEnable(GL\_DEPTH\_TEST);

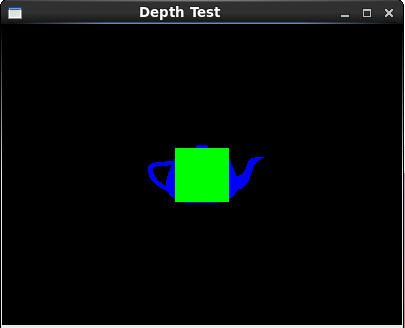
glutDisplayFunc(mydraw);

glutMainLoop();

return 0;

}

**OUTPUT:**

****

**PROGRAM NO: 14**

**AIM: Program to implement rendering.**

**PROGRAM:**

#include<GL/glu.h>

#include<GL/glut.h>

void init()

{

GLfloat mat\_specular[]={1.0,1.0,1.0,1.0};

GLfloat mat\_shininess[]={50.0};

GLfloat light\_position[]={1.0,1.0,1.0,0.0};

glClearColor(0.0,0.0,1.0,0.0);

glShadeModel(GL\_SMOOTH);

glMaterialfv(GL\_FRONT,GL\_SPECULAR,mat\_specular);

glMaterialfv(GL\_FRONT,GL\_SHININESS,mat\_shininess);

glLightfv(GL\_LIGHT0,GL\_POSITION,light\_position);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glutSolidSphere(1.0,20,16);

glFlush();

}

void reshape(int w,int h)

{

glViewport(0,0,(GLsizei)w,(GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

glOrtho(-1.5,1.5,-1.5\*(GLfloat)h/(GLfloat)w,1.5\*(GLfloat)h/(GLfloat)w,- 10.0,10.0);

else

glOrtho(-1.5\*(GLfloat)w/(GLfloat)h,1.5\*(GLfloat)w/(GLfloat)h,-1.5,1.5,-10.0,10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

int main(int argc,char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(500,500);

glutInitWindowPosition(100,100);

glutCreateWindow(argv[0]);

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**OUTPUT**:

