**CONSTRUCTORS AND DESTRUCTORS**

#include<iostream.h>

#include<conio.h>

class atm

{

public:

atm()

{

amount=5000;

}

~atm()

{

amount=0;

}

int amount;

void credit();

void debit();

void bal();

};

void atm::credit()

{

int c;

cout<<"enter a amount:";

cin>>c;

amount=amount+c;

cout<<"now your balance:"<<amount;

}

void atm::debit()

{

int d;

cout<<"enter a amount:";

cin>>d;

amount=amount-d;

cout<<"now your balance:"<<amount;

}

void atm::bal()

{

cout<<"your account balance:"<<amount;

}

void main()

{

atm ob;

int i,actno,pin[5],ch;

clrscr();

cout<<"enter your account number:";

cin>>actno;

cout<<"enter your pin:";

for(i=1;i<=5;i++)

{

pin[i]=getch();

cout<<"\*";

}

do

{

cout<<"\n1.credit\n2.debit\n3.balance\n4.exit\n";

cout<<"enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

ob.credit();

break;

case 2:

ob.debit();

break;

case 3:

ob.bal();

break;

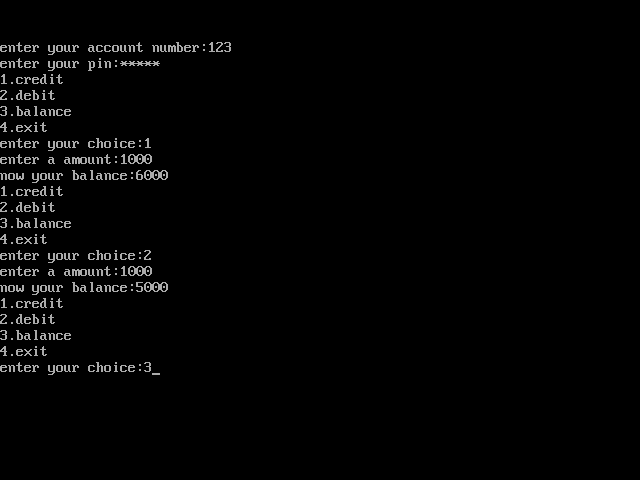
}

}while(ch!=4);

getch();

}

**OUTPUT:**



**STACK USING ARRAY**

#include<iostream.h>

#include<conio.h>

#define max 5

class stack

{

public:

int top;

int st[20];

void push(void);

void pop(void);

void display(void);

};

void stack::push()

{

if(top==max)

{

cout<<"\n stack is full";

}

else

{

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

top++;

cin>>st[top];

}

}

void stack::pop()

{

if(top==0)

{

cout<<"\n stack is empty";

}

else

{

cout<<"\n the deleted element is:"<<st[top];

top--;

}

}

void stack::display()

{

int i;

if(top==0)

{

cout<<"\n stack is empty";

}

else

{

cout<<"\n stack element are:\n";

for(i=1;i>=top;i++)

{

cout<<endl<<st[i];

}

}

}

void main()

{

stack ob;

int ch;

clrscr();

ob.top=0;

do

{

cout<<"\n1.push\n2.pop\n3.display\n4.exit\n";

cout<<"enter the choice";

cin>>ch;

switch(ch)

{

case 1:

ob.push();

break;

case 2:

ob.pop();

break;

case 3:

ob.display();

break;

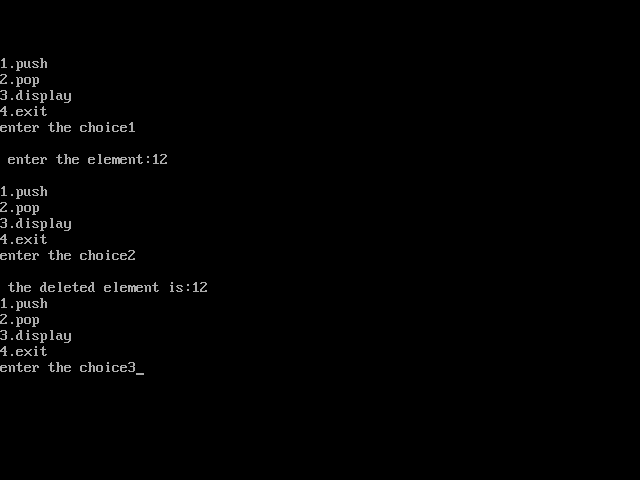
}

}while(ch!=4);

getch();

}

**OUTPUT:**



**QUEUE USING ARRAY**

#include<iostream.h>

#include<conio.h>

class queue

{

public:

int rear,front,q[10];

void enq(void);

void deq(void);

void display(void);

};

void queue::enq()

{

if(rear==5)

{

cout<<"\nqueue is full";

}

else if(front==0)

{

front=rear=1;

cout<<"\nenter the element:";

cin>>q[rear];

}

else

{

cout<<"\nenter the element:";

rear++;

cin>>q[rear];

}

}

void queue::deq()

{

if(front==0)

{

cout<<"\nqueue is empty";

}

else if(front==rear)

{

cout<<"\nthe deleted element is:"<<q[front];

front=rear=0;

}

else

{

cout<<"\nthe deleted element is:"<<q[front];

front++;

}

}

void queue::display()

{

int i;

if(rear==0)

{

cout<<"\nqueue is empty";

}

else

{

cout<<"\queue element are:\n";

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

{

cout<<endl<<q[i];

}

}

}

void main()

{

queue ob;

int ch;

clrscr();

ob.front=0;

ob.rear=0;

do

{

cout<<"\n1.enqueue\n2.dequeue\n3.display\n4.exit\n";

cout<<"enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

ob.enq();

break;

case 2:

ob.deq();

break;

case 3:

ob.display();

break;

}

}while(ch!=4);

getch();

}

**OUTPUT:**



**EXCEPTION HANDLING**

#include<iostream.h>

#include<conio.h>

void main()

{

int a,b,c;

clrscr();

cout<<"enter two no's";

cin>>a>>b;

try

{

if(b==0)

{

throw 1;

}

c=a/b;

cout<<c;

}

catch(int x)

{

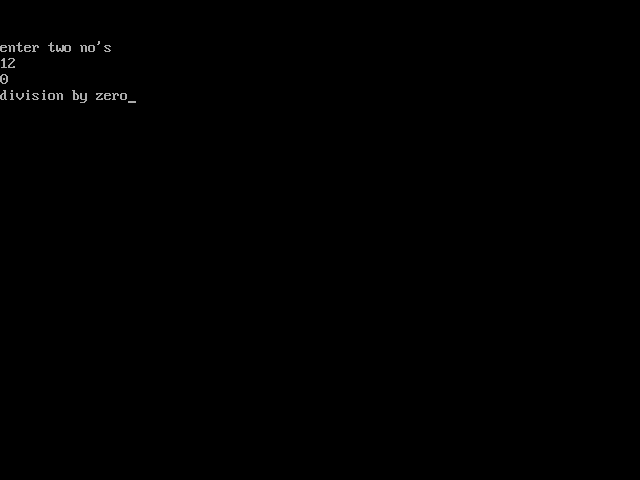
cout<<"division by zero";

}

getch();

}

**OUTPUT:**



**BINARY SEARCH**

#include<iostream.h>

#include<conio.h>

void main()

{

int a[5],i,n,t=0,beg=1,end=10,mid;

clrscr();

cout<<"enter the value:\n";

for(i=1;i<=5;i++)

{

cin>>a[i];

}

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

cin>>n;

while(beg<=end)

{

mid=(beg+end)/2;

if(n==a[mid])

{

cout<<"location:"<<mid;

t=1;

break;

}

else

{

if(n>a[mid])

{

beg=mid+1;

}

else

{

end=mid-1;

}

}

}

if(t==0)

{

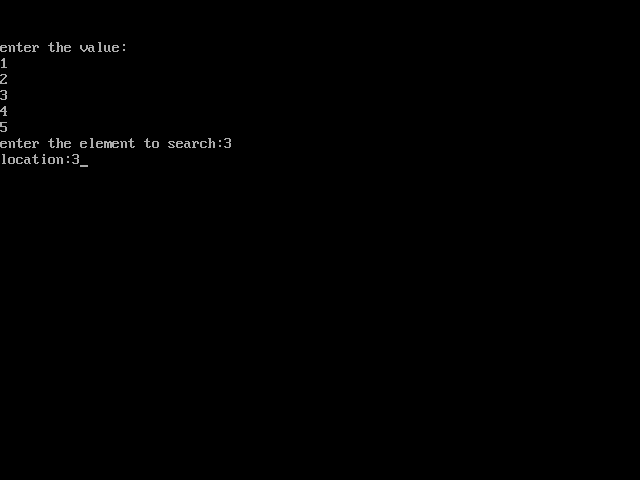
cout<<"element not found";

}

getch();

}

**OUTPUT:**



**BUBBLE SORT**

#include<iostream.h>

#include<conio.h>

void main()

{

int a[5],i,j,t,c=4;

clrscr();

cout<<"enter the value:\n";

for(i=1;i<=5;i++)

{

cin>>a[i];

}

for(i=1;i<5;i++)

{

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

{

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

{

t=a[j];

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

a[j+1]=t;

}

}

c--;

}

for(i=1;i<=5;i++)

{

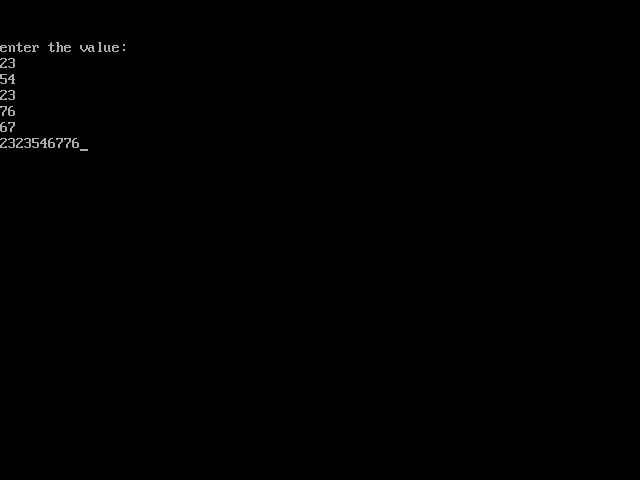
cout<<a[i];

}

getch();

}

**OUTPUT:**



**FUNCTION OVERLOADING**

#include<iostream.h>

#include<conio.h>

float area(float r)

{

float p=3.14;

return(p\*(r\*r));

}

float area(float l,float b)

{

return(l\*b);

}

void main()

{

float r=6.12,l=12.8,b=18.2;

clrscr();

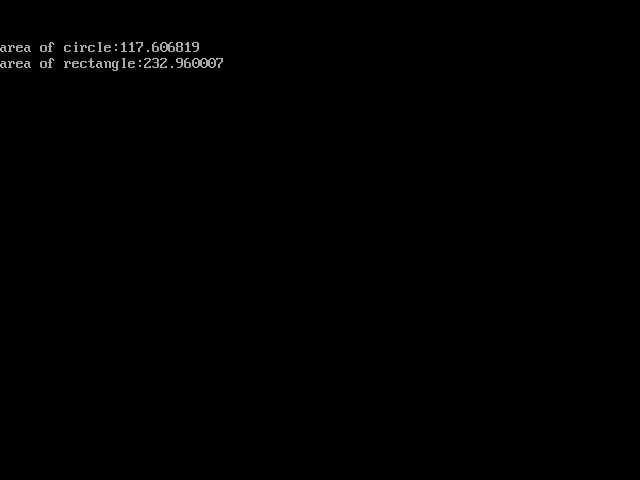
cout<<"area of circle:"<<area(r)<<endl;

cout<<"area of rectangle:"<<area(l,b)<<endl;

getch();

}

**OUTPUT:**



**INHERITANCE**

#include<iostream.h>

#include<conio.h>

class matrixa

{

public:

int a[3][3],i,j;

void getdata()

{

cout<<"enter a matrix:\n";

for(i=1;i<=2;i++)

{

for(j=1;j<=2;j++)

{

cin>>a[i][j];

}

}

}

};

class determination:public matrixa

{

public:

int d;

void fun1()

{

d=(a[1][1]\*a[2][2])-(a[1][2]\*a[2][1]);

cout<<"determinent of a:"<<d;

}

};

class rank:public determination

{

public:

void fun2()

{

if(d==0)

{

cout<<"\nrank of a: 1";

}

else

{

cout<<"\nrank of a: 2";

}

}

};

void main()

{

int ch;

clrscr();

determination det;

rank pa;

do

{

cout<<"\n1.determinent\n2.rank\n3.exit\n";

cout<<"enter your choice:";

cin>>ch;

switch(ch)

{

case 1:

det.getdata();

det.fun1();

break;

case 2:

pa.getdata();

pa.fun1();

pa.fun2();

break;

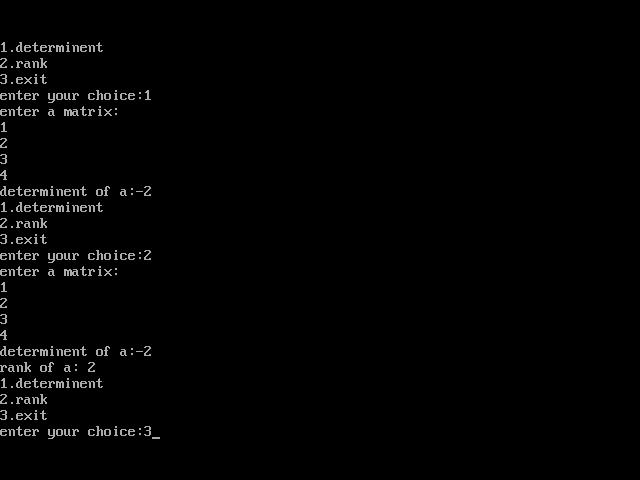
}

}while(ch!=3);

getch();

}

**OUTPUT:**



**OVERLOADING UNARY OPERATOR**

#include<iostream.h>

#include<conio.h>

class space

{

int x,y,z;

public:

void getdata(int a,int b,int c)

{

x=a;

y=b;

z=c;

}

void display()

{

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

}

void operator -()

{

x=-x;

y=-y;

z=-z;

}

};

void main()

{

space s;

clrscr();

s.getdata(10,-20,30);

s.display();

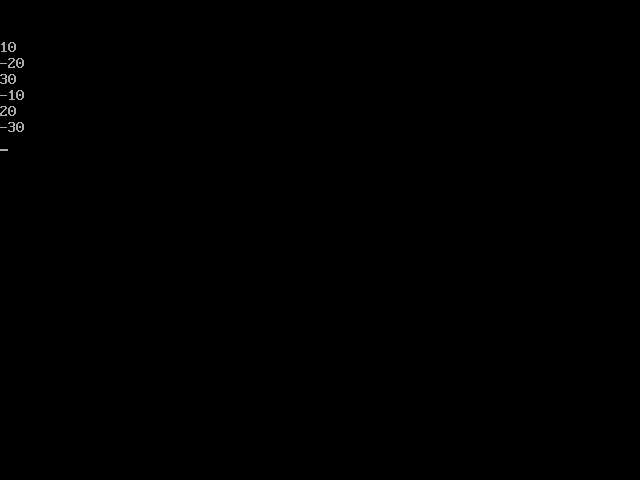
-s;

s.display();

getch();

}

**OUTPUT:**



**OVERLOADING BINARY OPERATOR**

#include<iostream.h>

#include<conio.h>

class complex

{

float x;

float y;

public:

complex()

{

}

complex(float real,float img)

{

x=real;

y=img;

}

complex operator+(complex);

void display()

{

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

}

};

complex complex :: operator+(complex c)

{

complex temp;

temp.x=x+c.x;

temp.y=y+c.y;

return(temp);

}

void main()

{

clrscr();

complex c1,c2,c3;

c1=complex(2.5,3.5);

c2=complex(1.6,2.7);

c3=c1+c2;

c1.display();

c2.display();

c3.display();

getch();

}

**OUTPUT:**

