```cpp
#include <iostream>
#include <cmath>
using namespace std;

class MiMatriz
{
    private:                                                    /* Declaramos las variables que vamos utilizar y que unicamente puede modificiar el programador de clase , en private */

        static const int MAX_FIL = 50 , MAX_COL = 50 ;      /* Como estamos creando una clase matriz , declaramos la matriz en privado y posteriormente crearemos modelos para su uso */
        int matriz_privada[MAX_FIL][MAX_COL];
        int util_fil;
        int util_col;
        int fila , columna ;

    public :

            MiMatriz(){                     /* Creamos un constructor , que al declarar una matriz en el main como MiMatriz , por defecto se inicialice la matriz en 0 , tanto filas como columnas */

            util_fil = 0 ;
            util_col = 0 ;
        }

    /*  MiMatriz(){         Tambien podiamos haber credo este constructor , que te genera una matriz de dos por dos que todos sus elementos valen 1


            util_fil = 2 ;
            util_col = 2 ;

            int i , j , k ;

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

                for ( j=0; j<= 4 ; j++){

                    for( k= 0 ; k <= 1 ; k++){

                        matriz_privada[i][j] = k ;
                    }
                }
            }

        } */

        int Dato(int fila, int columna)       /* Desde este módulo conseguimos que nos devuelva el valor del dato que pasaremos su fila y columna */
        {
```

```
            return matriz_privada[fila][columna];
        }

        void AsignarDato( int fila , int columna , int dato)        /*
Asignaremos un valor en un posicion determinada por una columna y fila */
            {
                matriz_privada[fila][columna] = dato ;
            }

        void Introd_Util_fil( int dato)                 /* Se asigna un valor
al ulti_fill que se utilizara en la matriz en posteriores operaciones */
            {
                util_fil = dato ;
            }

        void Introd_Util_col( int dato)     /* Al igual que con util_fill ,
haremos lo mismo con util_col , dandole un valor para trabajar con la matriz
*/
            {
                util_col = dato ;
            }

        int Total_col()                 /* Para facilitar el uso de distintos
modulos mas avanzados , utilizaremos este para devolver la cantidad de
columnas */
            {
                return util_col ;
            }

        int Total_fil()             /* Nos devulve el valor de la cantidad
total de fila */
            {
                return util_fil ;
            }

        void BorrarTodos()  /* Elimna los elementos de la matriz ,
devolviendola al estado que le dejó el constructor */
            {
                util_fil = 0;
                util_col = 0;
            }

        void Introd_Fila_ordenar ( int dato)  /* Se usa para facilitar el
modulo de ordenacion de fila , y te asigna a fila un valor dado sin devolver
nada */
            {
                fila = dato ;
            }

        int Fila_ordenar ()  /* De vuelve el valor asignado a la fila que
queremos ordenar */
            {
                return fila - 1 ;
            }
```

```
        void Introd_Columna_ordenar ( int dato)  /* Se usa para facilitar el
modulo de ordenacion de columna , y te asigna a fila un valor dado sin
devolver nada */
            {
                columna = dato ;
            }

        int Columna_ordenar ()  /* De vuelve el valor asignado a la columna
que queremos ordenar */
            {
                return columna - 1 ;
            }

        void OrdenarFila () /* Se realiza la ordenacion de la fila mediante
le medotodo de la burbuja */
        {
            bool cambio=true;
            int intercambia , dato1;

            for (int izda = 0; izda < Total_col() && cambio; izda++){  /* Se
utiliza Total_col ya declarada antes , para procesar la matriz con los for */
            cambio = false;
            for (int i = Total_col()-1 ; i > izda ; i--)
                {
                if (Dato(Fila_ordenar(),i) < Dato(Fila_ordenar(),i-1)){  /*
Aqui se utiliza el intercambio de datos para poder culminar el metedoo de
ordenacion por burbuja */
                    cambio = true;

                    intercambia = Dato(Fila_ordenar(),i);  // Se utiliza el
modulo dato para procesar la matriz e intercambiar los datos y ordenarlos , y
dentro de este se usa ,

                    dato1 = Dato(Fila_ordenar(),i-1);       // fila_ordenar ,
para ordenar la matriz

                    AsignarDato(Fila_ordenar(),i,dato1) ;

                    AsignarDato(Fila_ordenar(),i-1,intercambia) ; /* El
modulo de asignar dato , culmina el intercambio de los datos y su nueva
asignacion en la matriz */

                    }
                }
            }
        }

        void OrdenarColumna () /* Se realiza la ordenacion de la columna
mediante le medotodo de la burbuja */
        {
            bool cambio=true;
            int intercambia , dato1;
```

```
            for (int izda = 0; izda < Total_fil() && cambio; izda++){
            cambio = false;
            for (int i = Total_fil()-1 ; i > izda ; i--)
                {
                if (Dato(i,Columna_ordenar()) < Dato(i-1,Columna_ordenar())){

                    cambio = true;

                    intercambia = Dato(i,Columna_ordenar()) ;

                    dato1 = Dato(i-1,Columna_ordenar());

                    AsignarDato(i,Columna_ordenar(),dato1) ;

                    AsignarDato(i-1,Columna_ordenar(),intercambia) ;

                    }
                }
            }
        }

        void BuscaNumero (int entero_buscar )
        {                                                                   /* Este
modulo podriamos colocarlo tambien como funcion , y lo que hace es buscar si
esta entre la matriz el numero deseado a buscar */

            int fila , columna , i , j ;

            for ( i=0; i<util_fil ; i++) {

                for ( j=0; j<util_col ; j++){

                    if( matriz_privada[i][j] == entero_buscar ){       /* Si
encuentra el dato devuelve su posicion dada como fila y columna */

                        fila = i ;
                        columna = j ;
                    }
                }
            }

        }

   /*   void OrdenarBurbuja()
        {
            bool cambio=true;
            int intercambia , dato1 , j , i ,k , izda = 0 ,tmp ;

            while (cambio)
            cambio = false ;
            {
                for ( int i = util_fil - 1 ; i > izda  ; i--)
                {
                    for ( int j = util_col - 1 ; i > izda  ; j--)
```

```
                    {
                        if (matriz_privada[i][j] < matriz_privada[i - 1][j - 1])

                        tmp = matriz_privada[i][j] ;
                        matriz_privada[i][j] = matriz_privada[i - 1][j - 1 ] ;
                        matriz_privada[i - 1][j - 1 ] = tmp ;

                        cambio = true
                    }
                }
                izda ++ ;
            }

        } */
};

    MiMatriz CompletarMatriz (){  //Esta funcion lo que hace es procesar la matriz y completarla segun los datos del usuario , con sus filas , columnas y elemntos

    MiMatriz m ;     //Se declara la matriz con la clase ya realizaada arriba */
    int dato1, dato2;
    int valor ;

    cout<<"Introducir filas de la matriz: ";
    cin>>dato1 ;
    m.Introd_Util_fil(dato1);                    /*Se usa el metodo de la clase matriz declarada arriba , y se utiliza para introducir el tamaño del Util_fil y Util_col */

    cout<<"Introducir columnas de la matriz: ";
    cin>>dato2 ;
    m.Introd_Util_col(dato2);

        for(int i=0; i<m.Total_fil(); i++){     /* Para procesar la matriz es necesario conocer su total de filas y para ello se llama al modulo respectivo con el '.' */

            for(int j=0; j<m.Total_col(); j++){

            cout << "Introducir "<<j+1<<" elemento de la "<<i+1<<" fila: ";
            cin >>  valor ;

            m.AsignarDato(i,j,valor) ;     /*  Se llama al modulo para asignar dato en las respectivas columnas y filas */

            }
        }

        return m ;  /* Devuelve una matriz , ya que la funcion esta declarada como clase matriz */
```

```
    }

    void MostrarMatriz (MiMatriz m)  /* Es una funcion void que devuelve la
matriz de forma ordenada , tras haberla completado con la funcion de arriba */
        {
            cout<<"\n La matriz  es: \n"<<endl;

            for(int i=0;i<m.Total_fil();i++){   /* El void recibe como dato
una matriz declarada tipo MiMatriz que es la estructura realizada , para
hacer uso de sus distintos modulos */

                cout<<"| ";

                for(int j=0;j<m.Total_col();j++){

                    cout<<m.Dato(i,j)<<" ";         /* Se una el modulo de
matriz dato para ir mostrandolo mientras se procesa por filas y columnas */

                }
                cout<<"|"<<endl;
            }
        cout<<endl;
        }

    int MostrarMayor(MiMatriz m)  // Una funcion que muestra el mayor valor
de la matriz , para eso se le introduce a la funcion una matriz declarada
como MiMatriz , para hacer uso de modulos
    {

        int mayor=0; /* Inicializamos mayor como 0 , y en los bucles for
vamos comparandolo con 0 y si es mayor , ese valor se convierte ahora en
mayor , asi durante todo la matriz */

        for(int i=0; i<m.Total_fil(); i++){

            for(int j=0; j<m.Total_col(); j++){

                if(m.Dato(i,j)>mayor){

                    mayor=m.Dato(i,j);

                    }
                }
            }
            cout<<"El numero mayor es: "<<mayor<<endl;
            cout<<endl;
    }

    int MostrarMenor(MiMatriz m)
    {

        int menor=1;

        for(int i=0; i<m.Total_fil(); i++){
```

```
            for(int j=0; j<m.Total_col(); j++){

                if(m.Dato(i,j)<menor){

                    menor=m.Dato(i,j);

                    }
                }
            }
            cout<<"El numero menor es: "<<menor<<endl;
            cout<<endl;
    }

    MiMatriz IntrodVector (MiMatriz m ){   //Creo que es bastante explicito
este codigo , lo que realiza es introducir un vector en la matriz preguntando
al usuario por multiples opcciones operacionales

        const int MAX = 10;
        int v[MAX] , util_v , numero , vector  , select;

        cout << "\n Introducir tamaño del vector , no ha de ser mayor al
tamañado de filas o columnas(lo introduce en la matriz original): \n" ;
        cin >> util_v  ;

        util_v = util_v  ;

        for (int i=0; i<util_v; i++){

            cout << "\n Introducir dato del vector a introducir en matriz :
\n";
            cin >> v[i];
        }

        cout << "\n Introducir si se desea meter el vector en una fila(1) o
columna(2) : \n" ;
        cin >> select ;

        cout << "\n Introducir la fila o columna donde se desea introducir :
\n " ;
        cin >> numero ;

        numero = numero - 1 ;

        if ( select == 1){

            for(int i=0; i<util_v; i++){

                for(int j=0; j<=i; j++){

                    m.AsignarDato(numero,i,v[j]) ;

                }
            }
```

```
        }
                                                    //Al dar la opccion de por
filas o por columnas , si se elige de introducir en las filas, se procesa
solo las columnas y se va colocando el vector
        if ( select == 2){

            for(int i=0; i<m.Total_col(); i++){

                for(int j=0; j<=i; j++){

                    m.AsignarDato(i,numero,v[j]) ;

                }
            }
        }
        return m ;

    }

    void BuscaNumero (int entero_buscar, MiMatriz m)  /* Realiza lo mismo que
el modulo BuscaNumero */
    {

            bool encontrado ;
            int i , j , fila , columna ;

            for ( i=0; i<m.Total_fil() ; i++)
            {

                for ( j=0; j<m.Total_col() ; j++)
                {
                    if( m.Dato(i,j) == entero_buscar )

                        {
                            encontrado = true ;
                            fila = i ;
                            columna = j ;
                        }
                }
            }

            if ( encontrado )
            {
                cout << "El numero se encuentra en la posicion:  " << fila <<
"," << columna << endl ;
            }

            else

                cout << "No se encuentra el numero en la matriz" << endl ;
    }

    void FilUnicas(MiMatriz m){     /* Esta funcion te busca la cantidad de
filas unicas en la matriz , estas son las unicas que tienen unos valores no
repetidos en la matriz */
```

```
        int j , cu = 0 , i , k ;
        bool colrep , colunica ;

        for(j=0;j<m.Total_col();j++)  /* Recorremos primero la matriz desde
las filas y las columnas */
        {

        colunica=true;

            for(k=0;k<m.Total_col();k++)
            {

            colrep=true;

                for(i=0;i<m.Total_fil();i++)
                {

                if(m.Dato(i,j)!=m.Dato(i,k))   /* Si encuentras una columnas
con los valores iguales en otra columnas , se convierte en false */
                {
                    colrep=false;
                }
                if(colrep==true && j!=k )

                    {
                    colunica=false;
                    }
                }
            }
        if(colunica==true)   /* Si la columna es unica y no se encuentra otra
con los mismos valores , no se habra entrado en ningun buclue con false y
sera por tanto true y contara una */
            {
                cu++;
            }
        }
    cout<<"El numero de columnas unicas es: "<<cu;
    }

    void MatrizEspiral()   /* Crea una matriz cuyos nueros hacen una espiral
*/
    {
        int valor ;
        MiMatriz m ;

        do{
            cout << endl << "Introduzca el nº de filas de la matriz (impar >=
3): ";
            cin >> valor;
            m.Introd_Util_fil(valor) ;

        }while(m.Total_fil()%2==0 || m.Total_fil()<3);
```

```
        int cuadrado=pow(m.Total_fil(), 2);  /* Hacemos uso de la libreria de
matematicas de c++ para poder realizar todos los calculos que generaran la
matriz desde 1 hasta completar la espiral de valores */
        int centro=m.Total_fil()/2;         /* Ubicamos el centro de la
matriz de referencia como la cantidad de filas entre dos */
        int fila=centro;                    /* Asignamos tanto fila como
columna al valor de ese supuesto centro */
        int columna=centro;                          /* Cuadrado nos da la
cantidad de valores que hay en la matriz cuadrada */
        m.AsignarDato(fila,columna,cuadrado) ;   /* Por tanto el valor
centrar de la matriz , es decir el final ; valdra el cuadrado de la cantidad
de filas que tiene */
        cuadrado--;                               /* Restamos el valor de
cuadrado ahora a uno menos para le siguiente de la matriz */

        for(int i=1; i<=m.Total_fil(); i++)
        {
            if(i%2!=0)
            {
                for(int j=0; j<i; j++)              /* Vas recorriendo la
matriz de forma impar , restando el valor de columna , y asignadole el valor
de cuadrado que se va restando en uno cada interacion */
                {
                    columna--;
                    m.AsignarDato(fila,columna,cuadrado) ;      /* Por lo
tanto da como resueltado el recorrido en espiral de la matriz */
                    cuadrado--;
                }
                for(int j=0; j<i; j++)
                {
                    fila++;
                    m.AsignarDato(fila,columna,cuadrado) ;
                    cuadrado--;
                }
            }
            else                                              /* Aqui se
recorre la matriz con los pares tras no entrar en el if  , utilizando el
mismo modo de operacion */
            {
                for(int j=0; j<i; j++)
                {
                    columna++;
                    m.AsignarDato(fila,columna,cuadrado);
                    cuadrado--;
                }
                for(int j=0; j<i; j++)
                {
                    fila--;
                    m.AsignarDato(fila,columna,cuadrado);
                    cuadrado--;
                }
            }
        }
```

```
        cout << endl << "\nLa matriz es:\n"<<endl;   /*Tras crear la matriz y
tenerla lista en espiral , unicamente la mostramos por pantalla */
        for(int i=0; i<m.Total_fil(); i++)

        {
            cout << "\t";
            for(int j=0; j<m.Total_fil(); j++)
                cout<< m.Dato(i,j)<< " ";
            cout << endl;
        }


    }

    MiMatriz  intercambiar_filas(MiMatriz m)  /*Esta funcion intercambia las
filas dadas como argumento */
    {

        int tmp , fila1, fila2 ;

        cout << "\nIntroducir fila 1 a intercambiar : \n" ;
        cin >> fila1 ;

        cout << "Introducir fila 2 a intercambiar : " ;
        cin >> fila2 ;


        if ( fila1 > m.Total_fil() || fila2 > m.Total_fil())            /*
Realiza una comprobacion de que validez de las filas que le pasas a la
funcion */
        {
            cout << "Esta fuera de rango" << endl ;
        }

        else
/* Si las filas son correctas , se realiza el intercambio de sus valores , con
        {
            tmp ;
            fila1 -- ;
            fila2 -- ;
        }

        for ( int i =0 ;i < m.Total_col() ; i ++) {

            tmp = m.Dato(fila1,i) ;
            m.AsignarDato(fila1,i,m.Dato(fila2,i)) ;
            m.AsignarDato(fila2,i,tmp);
        }
        cout << "Se intercambian las filas : " << fila1+1 << "y" << fila2 + 1
<< endl;

        return m ;
    }
```

```
    MiMatriz Suma ( MiMatriz m1 , MiMatriz m2)      /* Las funciones suma ,
multiplicacion , resta y division , siguen el mismo planteamiento */
    {
        MiMatriz suma ;
        int dato ;

        if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() ==
m2.Total_col())
        {
            suma.Introd_Util_col(m1.Total_col()) ;
            suma.Introd_Util_fil(m1.Total_fil()) ;


            for ( int i = 0 ; i < m1.Total_fil() ; i ++)
            {
                for ( int j = 0 ; j < m1.Total_col() ; j ++)
                {
                    dato =  m1.Dato(i,j) + m2.Dato(i,j) ;
                    suma.AsignarDato(i,j,dato) ;
                }
            }
        }
        cout << "SUMA \n";
        return suma ;
    }

    MiMatriz Multiplicacion ( MiMatriz m1 , MiMatriz m2)
    {
        MiMatriz multiplicacion ;
        int dato ;

        if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() ==
m2.Total_col())
        {
            multiplicacion.Introd_Util_col(m1.Total_col()) ;
            multiplicacion.Introd_Util_fil(m1.Total_fil()) ;


            for ( int i = 0 ; i < m1.Total_fil() ; i ++)
            {
                for ( int j = 0 ; j < m1.Total_col() ; j ++)
                {
                    dato =  m1.Dato(i,j) * m2.Dato(i,j) ;
                    multiplicacion.AsignarDato(i,j,dato) ;
                }
            }
        }
        cout << "multiplicacion \n";
        return multiplicacion ;
    }

    MiMatriz Division ( MiMatriz m1 , MiMatriz m2)
    {
```

```
        MiMatriz Division ;
        int dato ;

        if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() ==
m2.Total_col())
        {
            Division.Introd_Util_col(m1.Total_col()) ;
            Division.Introd_Util_fil(m1.Total_fil()) ;


            for ( int i = 0 ; i < m1.Total_fil() ; i ++)
            {
                for ( int j = 0 ; j < m1.Total_col() ; j ++)
                {
                    dato =  m1.Dato(i,j) / m2.Dato(i,j) ;
                    Division.AsignarDato(i,j,dato) ;
                }
            }
        }
        cout << "Division \n" ;
        return Division ;
    }

    MiMatriz Resta ( MiMatriz m1 , MiMatriz m2)
    {
        MiMatriz Resta ;
        int dato ;

        if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() ==
m2.Total_col())
        {
            Resta.Introd_Util_col(m1.Total_col()) ;
            Resta.Introd_Util_fil(m1.Total_fil()) ;


            for ( int i = 0 ; i < m1.Total_fil() ; i ++)
            {
                for ( int j = 0 ; j < m1.Total_col() ; j ++)
                {
                    dato =  m1.Dato(i,j) - m2.Dato(i,j) ;
                    Resta.AsignarDato(i,j,dato) ;
                }
            }
        }
        cout << "Resta \n" ;
        return Resta ;
    }



int main () {

    int fil, col , numero , fila1 , fila2 ;                              /*
Para probar las posibilidades y capacidades de la clase MiMatriz creada , he
realizado un popurrí de ejemplos y ejercicios */
```

```
    MiMatriz matriz , m , m1 , m2;                                  /* Es
necesario declarar la matriz a usar como una de la clase MiMatriz */

    /* Si hubieras utilizado el constructor que genera la matriz 2x2,
podriamos haberla probaco con :   MostrarMatriz(constructor) ;  */


    matriz=CompletarMatriz();                           /* la matriz se pasa
a la funcion CompletarMatriz y usuario la rellena de datos e introduce sus
filas y columnas */

                                                    // Dicha funcion a su vez
utiliza los modulos aportados por la clase para facilitar y agilizar su usuo

    MostrarMatriz(matriz) ;                         // Le pasamos la matriz
a la funcion , que te la muestra por pantalla

    MostrarMayor(matriz);                           /* La funcion te muestra
el mayor valor de una matriz pasada como argumento */

    MostrarMenor(matriz);

    cout << "Introduzca fila de la matriz a ordenar: ";          /* Se le
pregunta al usuario que fila quiere ordenar */
    cin >> fil ;
    matriz.Introd_Fila_ordenar(fil) ;                           /* Se le
pasa al modulo que ordenada la fila de la matriz la fila introducida por le
usuario */

    matriz.OrdenarFila();
/* Se llama al modulo que ordenada las filas que esta relacionado con el modul

    MostrarMatriz(matriz) ;                                 /* Volvemos a
utilizar la funcion MostrarMatriz pero esta vez ya ordenada */

    cout << "Introduzca columna de la matriz a ordenar: ";
    cin >> col ;
    matriz.Introd_Columna_ordenar(col) ;

    matriz.OrdenarColumna();

    MostrarMatriz(matriz) ;

    cout << "Introduzca el numero a buscar en la matriz: " ;

argumento a la funcion que lo realiza */
    cin >> numero ;

    BuscaNumero(numero, matriz) ;

    FilUnicas (matriz) ;                            // Se llama a la
funcion FilUnica , para conocer cuantas filas unicas tiene la matriz
```

```
    MostrarMatriz(intercambiar_filas (matriz)) ;                    // Se
pasa como argumento , la natriz que devuelve la funcion con las filas ya
intercambiadas

    MostrarMatriz(IntrodVector(matriz)) ;                   //Se llama a la
funcion de IntrodVector para introducir el vector y dicha funcion devuelve
una matriz que muestra la funcion MostrarMatriz

    MatrizEspiral () ;                                                  //Se
llama a la funcion para generar la matriz espiral

    m1 = CompletarMatriz();
    m2 = CompletarMatriz();

    cout << "Las matrices por separado son : \n" << endl ;

    MostrarMatriz(m1) ;
    MostrarMatriz(m2) ;

    MostrarMatriz(Suma(m1,m2)) ;
    MostrarMatriz(Resta(m1,m2)) ;
    MostrarMatriz(Multiplicacion(m1,m2)) ;
    MostrarMatriz(Division(m1,m2)) ;

}

/* He realizado numerosas funciones en comparacion con los metodos de la
clase ya que
escuche que no se podia poner cin y cout en los metodo ( aunque esto se puede
resolver haciendo
un metodo en concreto de salidas y entrdas ) , no obstante , es muy facil
convertir las funciones en metodos
asi que su valoracion puede ser tanto como funcion o como metodo. */
```