

```

1: #include <iostream>
2: #include <cmath>
3: using namespace std;
4:
5: class MiMatriz
6: {
7:     private:                                     /* Declaramos las variables */
8:
9:         static const int MAX_FIL = 50 , MAX_COL = 50 ;      /* Como estamos creando una matriz de 50x50 */
10:        int matriz_privada[MAX_FIL][MAX_COL];
11:        int util_fil;
12:        int util_col;
13:        int fila , columna ;
14:
15:    public :
16:
17:        MiMatriz(){                                  /* Creamos un constructor , que al de */
18:
19:            util_fil = 0 ;
20:            util_col = 0 ;
21:        }
22:
23:        /* MiMatriz(){                                Tambien podiamos haber creado este constructor , q
24:
25:            util_fil = 2 ;
26:            util_col = 2 ;
27:
28:            int i , j , k ;
29:
30:            for ( i=0; i<= 4 ; i++) {
31:
32:                for ( j=0; j<= 4 ; j++){
33:
34:                    for( k= 0 ; k <= 1 ; k++){
35:
36:                        matriz_privada[i][j] = k ;
37:                    }
38:                }
39:            }
40:
41:        } */
42:
43:        int Dato(int fila, int columna)              /* Desde este módulo conseguimos qu
44:        {
45:            return matriz_privada[fila][columna];
46:        }
47:
48:        void AsignarDato( int fila , int columna , int dato)      /* Asignaremos el dato a la matriz */
49:        {
50:            matriz_privada[fila][columna] = dato ;
51:        }
52:
53:        void Introd_Util_fil( int dato)              /* Se asigna un valor al
54:        {

```

```

55:         util_fil = dato ;
56:     }
57:
58: void Introd_Util_col( int dato)      /* Al igual que con util_fill , harem
59: {
60:     util_col = dato ;
61: }
62:
63: int Total_col()                    /* Para facilitar el uso de distintos mo
64: {
65:     return util_col ;
66: }
67:
68: int Total_fil()                    /* Nos devuelve el valor de la cantidad total
69: {
70:     return util_fil ;
71: }
72:
73: void BorrarTodos() /* Elimna los elementos de la matriz , devolviendola
74: {
75:     util_fil = 0;
76:     util_col = 0;
77: }
78:
79: void Introd_Fila_ordenar ( int dato) /* Se usa para facilitar el modulo
80: {
81:     fila = dato ;
82: }
83:
84: int Fila_ordenar () /* De vuelve el valor asignado a la fila que queremos
85: {
86:     return fila - 1 ;
87: }
88:
89: void Introd_Columna_ordenar ( int dato) /* Se usa para facilitar el modu
90: {
91:     columna = dato ;
92: }
93:
94: int Columna_ordenar () /* De vuelve el valor asignado a la columna que q
95: {
96:     return columna - 1 ;
97: }
98:
99: void OrdenarFila () /* Se realiza la ordenacion de la fila mediante le me
100: {
101:     bool cambio=true;
102:     int intercambia , dato1;
103:
104:     for (int izda = 0; izda < Total_col() && cambio; izda++){ /* Se util
105:         cambio = false;
106:         for (int i = Total_col()-1 ; i > izda ; i--)
107:         {
108:             if (Dato(Fila_ordenar(),i) < Dato(Fila_ordenar(),i-1)){ /* Aqui

```

```

109:         cambio = true;
110:
111:         intercambia = Dato(Fila_ordenar(),i); // Se utiliza el modul
112:
113:         dato1 = Dato(Fila_ordenar(),i-1); // fila_ordenar , par
114:
115:         AsignarDato(Fila_ordenar(),i,dato1) ;
116:
117:         AsignarDato(Fila_ordenar(),i-1,intercambia) ; /* EL modulo de
118:
119:     }
120: }
121: }
122: }
123:
124: void OrdenarColumna () /* Se realiza la ordenacion de la columna mediante
125: {
126:     bool cambio=true;
127:     int intercambia , dato1;
128:
129:     for (int izda = 0; izda < Total_fil() && cambio; izda++){
130:         cambio = false;
131:         for (int i = Total_fil()-1 ; i > izda ; i--)
132:         {
133:             if (Dato(i,Columna_ordenar()) < Dato(i-1,Columna_ordenar())){
134:                 cambio = true;
135:
136:                 intercambia = Dato(i,Columna_ordenar()) ;
137:
138:                 dato1 = Dato(i-1,Columna_ordenar());
139:
140:                 AsignarDato(i,Columna_ordenar(),dato1) ;
141:
142:                 AsignarDato(i-1,Columna_ordenar(),intercambia) ;
143:             }
144:         }
145:     }
146: }
147:
148: void BuscaNumero (int entero_buscar )
149: {
150:
151:     int fila , columna , i , j ;
152:
153:     for ( i=0; i<util_fil ; i++) {
154:
155:         for ( j=0; j<util_col ; j++){
156:
157:             if( matriz_privada[i][j] == entero_buscar ){ /* Si encu
158:
159:                 fila = i ;
160:                 columna = j ;
161:             }
162:         }

```

```

163:         }
164:
165:     }
166:
167:     /* void OrdenarBurbuja()
168:     {
169:         bool cambio=true;
170:         int intercambia , dato1 , j , i ,k , izda = 0 ,tmp ;
171:
172:         while (cambio)
173:             cambio = false ;
174:         {
175:             for ( int i = util_fil - 1 ; i > izda ; i--)
176:             {
177:                 for ( int j = util_col - 1 ; i > izda ; j--)
178:                 {
179:                     if (matriz_privada[i][j] < matriz_privada[i - 1][
180:
181:                         tmp = matriz_privada[i][j] ;
182:                         matriz_privada[i][j] = matriz_privada[i - 1][j -
183:                         matriz_privada[i - 1][j - 1 ] = tmp ;
184:
185:                         cambio = true
186:                     }
187:                 }
188:                 izda ++ ;
189:             }
190:
191:         } */
192:     };
193:
194:     MiMatriz CompletarMatriz () { //Esta funcion lo que hace es procesar la matriz
195:
196:         MiMatriz m ; //Se declara la matriz con la clase ya realizaada arriba */
197:         int dato1,dato2;
198:         int valor ;
199:
200:         cout<<"Introducir filas de la matriz: ";
201:         cin>>dato1 ;
202:         m.Introd_Util_fil(dato1); //Se usa el metodo de la clase ma
203:
204:         cout<<"Introducir columnas de la matriz: ";
205:         cin>>dato2 ;
206:         m.Introd_Util_col(dato2);
207:
208:         for(int i=0; i<m.Total_fil(); i++){ // Para procesar la matriz es nec
209:
210:             for(int j=0; j<m.Total_col(); j++){
211:
212:                 cout << "Introducir "<<j+1<<" elemento de la "<<i+1<<" fila: ";
213:                 cin >> valor ;
214:
215:                 m.AsignarDato(i,j,valor) ; // Se llama al modulo para asignar da
216:

```

```

217:         }
218:     }
219:
220:     return m ; /* Devuelve una matriz , ya que la funcion esta declarada como
221:
222: }
223:
224: void MostrarMatriz (MiMatriz m) /* Es una funcion void que devuelve la matriz
225: {
226:     cout<<"\n La matriz es: \n"<<endl;
227:
228:     for(int i=0;i<m.Total_fil();i++){ /* El void recibe como dato una matriz
229:
230:         cout<<"| ";
231:
232:         for(int j=0;j<m.Total_col();j++){
233:
234:             cout<<m.Dato(i,j)<<" "; /* Se usa el modulo de matriz
235:
236:         }
237:         cout<<"| "<<endl;
238:     }
239:     cout<<endl;
240: }
241:
242: int MostrarMayor(MiMatriz m) // Una funcion que muestra el mayor valor de la
243: {
244:
245:     int mayor=0; /* Inicializamos mayor como 0 , y en los bucles for vamos con
246:
247:     for(int i=0; i<m.Total_fil(); i++){
248:
249:         for(int j=0; j<m.Total_col(); j++){
250:
251:             if(m.Dato(i,j)>mayor){
252:
253:                 mayor=m.Dato(i,j);
254:
255:             }
256:         }
257:     }
258:     cout<<"El numero mayor es: "<<mayor<<endl;
259:     cout<<endl;
260: }
261:
262: MiMatriz IntrodVector (MiMatriz m ){ /*Creo que es bastante explicito este
263:
264:     const int MAX = 10;
265:     int v[MAX] , util_v , numero , vector , select;
266:
267:     cout << "\n Introducir tamaño del vector , no ha de ser mayor al tamaño de
268:     cin >> util_v ;
269:
270:     util_v = util_v ;

```

```

271:
272:     for (int i=0; i<util_v; i++){
273:
274:         cout << "\n Introducir dato del vector a introducir en matriz : \n";
275:         cin >> v[i];
276:     }
277:
278:     cout << "\n Introducir si se desea meter el vector en una fila(1) o columna(2) : \n";
279:     cin >> select ;
280:
281:     cout << "\n Introducir la fila o columna donde se desea introducir : \n ";
282:     cin >> numero ;
283:
284:     numero = numero - 1 ;
285:
286:     if ( select == 1){
287:
288:         for(int i=0; i<util_v; i++){
289:
290:             for(int j=0; j<=i; j++){
291:
292:                 m.AsignarDato(numero,i,v[j]) ;
293:
294:             }
295:         }
296:     }
297:
298:     //Al dar la opcion de por filas
299:     if ( select == 2){
300:
301:         for(int i=0; i<m.Total_col(); i++){
302:
303:             for(int j=0; j<=i; j++){
304:
305:                 m.AsignarDato(i,numero,v[j]) ;
306:
307:             }
308:         }
309:         return m ;
310:
311:     }
312:
313: void BuscaNumero (int entero_buscar, MiMatriz m) /* Realiza lo mismo que el
314: {
315:
316:     bool encontrado ;
317:     int i , j , fila , columna ;
318:
319:     for ( i=0; i<m.Total_fil() ; i++)
320:     {
321:
322:         for ( j=0; j<m.Total_col() ; j++)
323:         {
324:             if( m.Dato(i,j) == entero_buscar )

```

```

325:
326:         {
327:             encontrado = true ;
328:             fila = i ;
329:             columna = j ;
330:         }
331:     }
332: }
333:
334:     if ( encontrado )
335:     {
336:         cout << "El numero se encuentra en la posicion: " << fila << ", "
337:     }
338:
339:     else
340:
341:         cout << "No se encuentra el numero en la matriz" << endl ;
342: }
343:
344: void FilUnicas(MiMatriz m){ /* Esta funcion te busca la cantidad de filas */
345:
346:     int j , cu = 0 , i , k ;
347:     bool colrep , colunica ;
348:
349:     for(j=0;j<m.Total_col();j++) /* Recorremos primero la matriz desde las f
350:     {
351:
352:         colunica=true;
353:
354:         for(k=0;k<m.Total_col();k++)
355:         {
356:
357:             colrep=true;
358:
359:             for(i=0;i<m.Total_fil();i++)
360:             {
361:
362:                 if(m.Dato(i,j)!=m.Dato(i,k)) /* Si encuentras una columnas con
363:                 {
364:                     colrep=false;
365:                 }
366:                 if(colrep==true && j!=k )
367:
368:                     {
369:                         colunica=false;
370:                     }
371:             }
372:         }
373:         if(colunica==true) /* Si la columna es unica y no se encuentra otra con
374:         {
375:             cu++;
376:         }
377:     }
378:     cout<<"El numero de columnas unicas es: "<<cu;

```

```

379:     }
380:
381: void MatrizEspiral()    /* Crea una matriz cuyos nueros hacen una espiral */
382: {
383:     int valor ;
384:     MiMatriz m ;
385:
386:     do{
387:         cout << endl << "Introduzca el nº de filas de la matriz (impar >= 3):";
388:         cin >> valor;
389:         m.Introd_Util_fil(valor) ;
390:
391:     }while(m.Total_fil()%2==0 || m.Total_fil()<3);
392:
393:     int cuadrado=pow(m.Total_fil(), 2); /* Hacemos uso de la libreria de mat
394:     int centro=m.Total_fil()/2;         /* Ubicamos el centro de la matriz de
395:     int fila=centro;                     /* Asignamos tanto fila como columna
396:     int columna=centro;                  /* Cuadrado nos da la cantida
397:     m.AsignarDato(fila,columna,cuadrado) ; /* Por tanto el valor centrar de
398:     cuadrado--;                          /* Restamos el valor de cuadrado a
399:
400:     for(int i=1; i<=m.Total_fil(); i++)
401:     {
402:         if(i%2!=0)
403:         {
404:             for(int j=0; j<i; j++)          /* Vas recorriendo la matriz
405:             {
406:                 columna--;
407:                 m.AsignarDato(fila,columna,cuadrado) ; /* Por lo tanto a
408:                 cuadrado--;
409:             }
410:             for(int j=0; j<i; j++)
411:             {
412:                 fila++;
413:                 m.AsignarDato(fila,columna,cuadrado) ;
414:                 cuadrado--;
415:             }
416:         }
417:         else                                /* Aqui se recorre la
418:         {
419:             for(int j=0; j<i; j++)
420:             {
421:                 columna++;
422:                 m.AsignarDato(fila,columna,cuadrado);
423:                 cuadrado--;
424:             }
425:             for(int j=0; j<i; j++)
426:             {
427:                 fila--;
428:                 m.AsignarDato(fila,columna,cuadrado);
429:                 cuadrado--;
430:             }
431:         }
432:     }

```



```

433:
434:     cout << endl << "\nLa matriz es:\n"<<endl;    /*Tras crear la matriz y ten
435:     for(int i=0; i<m.Total_fil(); i++)
436:
437:     {
438:         cout << "\t";
439:         for(int j=0; j<m.Total_fil(); j++)
440:             cout<< m.Dato(i,j)<< " ";
441:         cout << endl;
442:     }
443:
444:
445: }
446:
447: MiMatriz intercambiar_filas(MiMatriz m)    /*Esta funcion intercambia las fila
448: {
449:
450:     int tmp , fila1, fila2 ;
451:
452:     cout << "\nIntroducir fila 1 a intercambiar : \n" ;
453:     cin >> fila1 ;
454:
455:     cout << "Introducir fila 2 a intercambiar : " ;
456:     cin >> fila2 ;
457:
458:
459:     if ( fila1 > m.Total_fil() || fila2 > m.Total_fil())    /* Realiz
460:     {
461:         cout << "Esta fuera de rango" << endl ;
462:     }
463:
464:     else    /* Si
465:     {
466:         tmp ;
467:         fila1 -- ;
468:         fila2 -- ;
469:     }
470:
471:     for ( int i =0 ;i < m.Total_col() ; i ++ ) {
472:
473:         tmp = m.Dato(fila1,i) ;
474:         m.AsignarDato(fila1,i,m.Dato(fila2,i)) ;
475:         m.AsignarDato(fila2,i,tmp);
476:     }
477:     cout << "Se intercambian las filas : " << fila1+1 << "y" << fila2 + 1 <<
478:
479:     return m ;
480: }
481:
482: MiMatriz Suma ( MiMatriz m1 , MiMatriz m2)    /* Las funciones suma , multi
483: {
484:     MiMatriz suma ;
485:     int dato ;
486:

```

```

487:         if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() == m2.Total_col())
488:         {
489:             suma.Introd_Util_col(m1.Total_col()) ;
490:             suma.Introd_Util_fil(m1.Total_fil()) ;
491:
492:
493:             for ( int i = 0 ; i < m1.Total_fil() ; i ++)
494:             {
495:                 for ( int j = 0 ; j < m1.Total_col() ; j ++)
496:                 {
497:                     dato = m1.Dato(i,j) + m2.Dato(i,j) ;
498:                     suma.AsignarDato(i,j,dato) ;
499:                 }
500:             }
501:         }
502:         cout << "SUMA \n"
503:         return suma ;
504:     }
505:
506: MiMatriz Multiplicacion ( MiMatriz m1 , MiMatriz m2)
507: {
508:     MiMatriz multiplicacion ;
509:     int dato ;
510:
511:     if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() == m2.Total_col())
512:     {
513:         multiplicacion.Introd_Util_col(m1.Total_col()) ;
514:         multiplicacion.Introd_Util_fil(m1.Total_fil()) ;
515:
516:
517:         for ( int i = 0 ; i < m1.Total_fil() ; i ++)
518:         {
519:             for ( int j = 0 ; j < m1.Total_col() ; j ++)
520:             {
521:                 dato = m1.Dato(i,j) * m2.Dato(i,j) ;
522:                 multiplicacion.AsignarDato(i,j,dato) ;
523:             }
524:         }
525:     }
526:     cout << "multiplicacion \n"
527:     return multiplicacion ;
528: }
529:
530: MiMatriz Division ( MiMatriz m1 , MiMatriz m2)
531: {
532:     MiMatriz Division ;
533:     int dato ;
534:
535:     if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() == m2.Total_col())
536:     {
537:         Division.Introd_Util_col(m1.Total_col()) ;
538:         Division.Introd_Util_fil(m1.Total_fil()) ;
539:
540:

```

```

541:         for ( int i = 0 ; i < m1.Total_fil() ; i ++ )
542:         {
543:             for ( int j = 0 ; j < m1.Total_col() ; j ++ )
544:             {
545:                 dato = m1.Dato(i,j) / m2.Dato(i,j) ;
546:                 Division.AsignarDato(i,j,dato) ;
547:             }
548:         }
549:     }
550:     cout << "Division \n"
551:     return Division ;
552: }
553:
554: MiMatriz Resta ( MiMatriz m1 , MiMatriz m2)
555: {
556:     MiMatriz Resta ;
557:     int dato ;
558:
559:     if(m1.Total_fil() == m2.Total_fil() && m1.Total_col() == m2.Total_col())
560:     {
561:         Resta.Introd_Util_col(m1.Total_col()) ;
562:         Resta.Introd_Util_fil(m1.Total_fil()) ;
563:
564:
565:         for ( int i = 0 ; i < m1.Total_fil() ; i ++ )
566:         {
567:             for ( int j = 0 ; j < m1.Total_col() ; j ++ )
568:             {
569:                 dato = m1.Dato(i,j) - m2.Dato(i,j) ;
570:                 Resta.AsignarDato(i,j,dato) ;
571:             }
572:         }
573:     }
574:     cout << "Resta \n"
575:     return Resta ;
576: }
577:
578:
579:
580: int main () {
581:
582:     int fil, col , numero , fila1 , fila2 ;
583:     MiMatriz matriz , m , m1 , m2;
584:
585:     /* Si hubieras utilizado el constructor que genera la matriz 2x2, podriam
586:
587:
588:     matriz=CompletarMatriz();
589:
590:     /* La matriz se pasa a La
591:
592:     // Dicha funcion a su vez uti
593:
594:     // Le pasamos la matriz a La
595:
596:     /* La funcion te muestra el m

```

```

595:
596:     cout << "Introduzca fila de la matriz a ordenar: ";           /* Se le preg
597:     cin >> fil ;
598:     matriz.Introd_Fila_ordenar(fil) ;                               /* Se le pasa
599:
600:     matriz.OrdenarFila();                                           /* Se
601:
602:     MostrarMatriz(matriz) ;                                         /* Volvemos a utiliz
603:
604:     cout << "Introduzca column de la matriz a ordenar: ";
605:     cin >> col ;
606:     matriz.Introd_Columna_ordenar(col) ;
607:
608:     matriz.OrdenarColumna();
609:
610:     MostrarMatriz(matriz) ;
611:
612:     cout << "Introduzca el numero a buscar en la matriz: " ;
613:     cin >> numero ;
614:
615:     BuscaNumero(numero, matriz) ;
616:
617:     FilUnicas (matriz) ;                                           // Se llama a la funcion
618:
619:     MostrarMatriz(intercambiar_filas (matriz)) ;                   // Se pasa co
620:
621:     MostrarMatriz(IntrodVector(matriz)) ;                           //Se llama a la funci
622:
623:     MatrizEspiral () ;                                           //Se llama a
624:
625:     m1 = CompletarMatriz();
626:     m2 = CompletarMatriz();
627:
628:     cout << "Las matrices por separado son : \n" << endl ;
629:
630:     MostrarMatriz(m1) ;
631:     MostrarMatriz(m2) ;
632:
633:     MostrarMatriz(Suma(m1,m2)) ;
634:     MostrarMatriz(Resta(m1,m2)) ;
635:     MostrarMatriz(Multiplicacion(m1,m2)) ;
636:     MostrarMatriz(Division(m1,m2)) ;
637:
638: }
639:
640: /* He realizado numerosas funciones en comparacion con Los metodos de La
641: escuche que no se podia poner cin y cout en Los metodo ( aunque esto se p
642: un metodo en concreto de salidas y entrdsas ) , no obstante , es muy facil
643: asi que su valoracion puede ser tanto como funcion o como metodo. */

```