```
1   //---------------------------------------------------------------------
2
3   #pragma hdrstop
4
5   //---------------------------------------------------------------------
6
7   #pragma argsused
8   #include <cstdlib>
9   #include <iostream>
10  #include <iomanip>
11
12  using namespace std;
13
14  typedef unsigned short ushort;
15  typedef float *floatptr;
16  typedef float **floatmat;
17
18  class floatmatriz
19  {
20       private:
21            float **data;
22            ushort    fil,col;
23
24            void asignar (float **pData);
25            void asignar (floatmatriz& pMat);
26            void dimensionar (ushort pFil,ushort pCol);
27
28       public:
29            floatmatriz(ushort pFil,ushort pCol);
30            floatmatriz(ushort pFil);
31            floatmatriz(floatmatriz& pMat);
32            ~floatmatriz();
33
34            floatmatriz& operator = (floatmatriz& pMat);
35            floatmatriz& operator = (float pVal);
36            floatmatriz& operator + (floatmatriz& pMat);
37            floatmatriz& operator - (floatmatriz& pMat);
38            floatmatriz& operator - (void);
39            floatmatriz& operator * (floatmatriz& pMat);
40            floatmatriz& operator * (float pVal);
41            floatmatriz& operator / (floatmatriz& pMat);      // Pendiente por implementación
42            floatptr&    operator [] (ushort pFil);
43            floatmatriz& transpuesta (void);
44            floatmatriz& triangularsuperior (void);
45            floatmatriz& triangularinferior (void);
46            floatmatriz& inversa (void);                      // Pendiente por implementación
47            float        determinante (void);
48            void         aleatorio(void);
49
50            bool operator == (floatmatriz& pMat);
51            friend ostream& operator << (ostream& pOs,floatmatriz& pMat);
52  };
53
54  void floatmatriz::asignar (float **pData)
```

```
55  {
56       ushort f,c;
57
58       for (f=0;f<fil;f++)
59            for (c=0;c<col;c++)
60                 data[f][c]=pData[f][c];
61  }
62
63  void floatmatriz::asignar (floatmatriz& pMat)
64  {
65       ushort f,c;
66
67       if ((pMat.fil==fil)&&(pMat.col==col))
68          asignar(pMat.data);
69  }
70
71  void floatmatriz::dimensionar (ushort pFil,ushort pCol)
72  {
73       ushort f;
74
75       fil= pFil;
76       col= pCol;
77
78       data= new floatptr[fil];
79
80       for (f=0;f<fil;f++)
81            data[f]= new float[col];
82  }
83
84  floatmatriz::floatmatriz (ushort pFil,ushort pCol)
85  {
86          dimensionar(pFil,pCol);
87  }
88
89  floatmatriz::floatmatriz (ushort pFil)
90  {
91       dimensionar(pFil,pFil);
92  }
93
94  floatmatriz::floatmatriz(floatmatriz& pMat)
95  {
96       dimensionar(pMat.fil,pMat.col);
97       asignar(pMat);
98  }
99
100 floatmatriz::~floatmatriz ()
101 {
102      ushort f;
103
104      for (f=0;f<fil;f++)
105           delete data[f];
106
107      delete data;
108 }
```

```
109
110  floatmatriz& floatmatriz::operator = (floatmatriz& pMat)
111  {
112       asignar(pMat);
113       return *this;
114  }
115
116  floatmatriz& floatmatriz::operator = (float pVal)
117  {
118       ushort f,c;
119
120       for (f=0;f<fil;f++)
121            for (c=0;c<col;c++)
122                 data[f][c]= pVal;
123
124       return *this;
125  }
126
127  floatmatriz& floatmatriz::operator + (floatmatriz& pMat)
128  {
129       ushort f,c;
130       static floatmatriz *T= NULL;
131
132       if (T!=NULL)
133            delete T;
134       T= new floatmatriz(fil,col);
135
136       if ((fil==pMat.fil)&&(col==pMat.col))
137       {
138            for (f=0;f<fil;f++)
139                 for (c=0;c<col;c++)
140                      T->data[f][c]= data[f][c]+pMat.data[f][c];
141       }
142       else T= this;
143
144       return *T;
145  }
146
147  floatmatriz& floatmatriz::operator - (floatmatriz& pMat)
148  {
149       ushort f,c;
150       static floatmatriz *T= NULL;
151
152       if (T!=NULL)
153            delete T;
154       T= new floatmatriz(fil,col);
155
156       if ((fil==pMat.fil)&&(col==pMat.col))
157            for (f=0;f<fil;f++)
158                 for (c=0;c<col;c++)
159                      T->data[f][c]= data[f][c]-pMat.data[f][c];
160       else T= this;
161
162       return *T;
```

```
163  }
164
165  floatmatriz& floatmatriz::operator - (void)
166  {
167       ushort f,c;
168       static floatmatriz *T= NULL;
169
170       if (T!=NULL)
171            delete T;
172       T= new floatmatriz(fil,col);
173
174       for (f=0;f<fil;f++)
175            for (c=0;c<col;c++)
176                 T->data[f][c]*= -data[f][c];
177
178       return *T;
179  }
180
181  floatmatriz& floatmatriz::operator * (floatmatriz& pMat)
182  {
183       ushort f,c,oc,n;
184       static floatmatriz *T= NULL;
185
186       if (T!=NULL)
187            delete T;
188       T= new floatmatriz(fil,pMat.col);
189       *T=0;
190
191       if (col==pMat.fil)
192       {
193            for (f=0;f<T->fil;f++)
194                 for (c=0;c<T->col;c++)
195                      for (n=0;n<col;n++)
196                           T->data[f][c]+= data[f][n]*pMat.data[n][c];
197       }
198
199       return *T;
200  }
201
202  floatmatriz& floatmatriz::operator * (float pVal)
203  {
204       ushort f,c;
205       static floatmatriz *T= NULL;
206
207       if (T!=NULL)
208            delete T;
209       T= new floatmatriz(fil,col);
210
211       for (f=0;f<fil;f++)
212            for (c=0;c<col;c++)
213                 T->data[f][c]*= pVal;
214
215       return *T;
216  }
```

```
217
218  floatptr& floatmatriz::operator [] (ushort pFil)
219  {
220       return data[pFil];
221  }
222
223  floatmatriz& floatmatriz::transpuesta (void)
224  {
225       ushort f,c;
226       static floatmatriz *T= NULL;
227
228       if (T!=NULL)
229            delete T;
230       T= new floatmatriz(col,fil);
231
232       for (f=0;f<fil;f++)
233            for (c=0;c<col;c++)
234                 T->data[c][f]= data[f][c];
235
236       return *T;
237  }
238
239  floatmatriz& floatmatriz::triangularsuperior (void)
240  {
241       ushort n,f,c;
242       float p;
243       static floatmatriz *T= NULL;
244
245       if (T!=NULL)
246            delete T;
247
248       if (fil == col)
249       {
250            T= new floatmatriz(*this);
251
252            for (n=0;n<fil;n++)
253                 for (f=n+1;f<fil;f++)
254                      for (c=0,p=T->data[f][n];c<col;c++)
255                          T->data[f][c]= T->data[f][c]-p/T->data[n][n]*T->data[n][c];
256       }
257
258       return *T;
259  }
260
261  floatmatriz& floatmatriz::triangularinferior (void)
262  {
263       short n,f,c;
264       float p;
265       static floatmatriz *T= NULL;
266
267       if (T!=NULL)
268            delete T;
269
270       if (fil == col)
```

```
271         {
272              T= new floatmatriz(*this);
273
274              for (n=fil-1;n>=0;n--)
275                   for (f=n-1;f>=0;f--)
276                        for (c=col-1,p=T->data[f][n];c>=0;c--)
277                             T->data[f][c]= T->data[f][c]-p/T->data[n][n]*T->data[n][c];
278         }
279
280         return *T;
281    }
282
283    float floatmatriz::determinante (void)
284    {
285            ushort n;
286            floatmatriz T(fil,col);
287            float det= 0;
288
289            if (fil==col)
290            {
291                    det= 1;
292                    T= triangularsuperior();
293                    for (n=0;n<fil;n++)
294                            det*= T.data[n][n];
295            }
296
297            return det;
298    }
299
300    bool floatmatriz::operator == (floatmatriz& pMat)
301    {
302         ushort f,c;
303
304         if ((fil==pMat.fil)&&(col==pMat.col))
305         {
306              for (f=0;f<fil;f++)
307                   for (c=0;c<col;c++)
308                        if (data[f][c]!=pMat.data[f][c])
309                             return false;
310              return true;
311         }
312
313         return false;
314    }
315
316    void floatmatriz::aleatorio (void)
317    {
318         static bool primeraVez= true;
319         ushort f,c;
320
321         if (primeraVez)
322         {
323              srand((unsigned)time(0));
324              primeraVez= false;
```

```
325         }
326
327         for (f=0;f<fil;f++)
328              for (c=0;c<col;c++)
329                   data[f][c]= rand()/100.0;
330    }
331
332    ostream& operator << (ostream& pOs,floatmatriz& pMat)
333    {
334         ushort f,c;
335
336         for (f=0;f<pMat.fil;f++)
337         {
338              for (c=0;c<pMat.col;c++)
339                   pOs << pMat.data[f][c] << "\t";
340              pOs << endl;
341         }
342
343         return pOs;
344    }
345
346    int main(void)
347    {
348
349         floatmatriz A(2,3),B(3,2),C(2,3),D(3,3),E(A);
350
351         A[0][0]= 2;
352         A[0][1]= 4;
353         A[0][2]= 6;
354         A[1][0]= -1;
355         A[1][1]= 3;
356         A[1][2]= 9;
357         cout << A << endl;
358
359         B[0][0]= -5;
360         B[1][0]= -7;
361         B[2][0]= 6;
362         B[0][1]= -5;
363         B[1][1]= -7;
364         B[2][1]= 6;
365         cout << B << endl;
366
367         cout << A*B << endl;
368
369         C=20.5;
370
371         cout << C << endl;
372
373         if (A==C)
374              cout << "iguales";
375         else cout << "diferentes";
376         cout << endl;
377
378         D[0][0]= 166;
```

```
379         D[0][1]= 244;
380         D[0][2]= 3554;
381         D[1][0]= 444;
382         D[1][1]= 5543;
383         D[1][2]= 6654;
384         D[2][0]= 7645;
385         D[2][1]= -8654;
386         D[2][2]= 9243;
387
388         cout.precision(2);
389         cout << right << scientific << showpos << setw(8) << setfill(' ');
390
391         cout << D << endl << endl;
392         cout << D.triangularsuperior() << endl;
393         cout << D.triangularinferior() << endl;
394         cout << "Determinante: " << D.determinante() << endl;
395
396         system("PAUSE");
397         return EXIT_SUCCESS;
398     }
399
400     //--------------------------------------------------------------------
401
402
```