```
 1: program TrabalhoComputacional2_VersaoFinal;
 2:
 3: uses crt;
 4:
 5: var  operacao   :        Integer;         //Operações com matrizes
 6:      op         :        Integer;         //Operando geral para captação no case
 7:      k          :        Integer;         //i,j e k - contadores dos lacos for
 8:      i          :        Integer;
 9:      j          :        Integer;
10:      vr         :        Integer;         //vr - valor constante p/ operação 3
11:      vet1       :        Array [1..2] of Integer;
12:      vet2       :        Array [1..2] of Integer;     // Vet1 a Vet7 - vetores
13:      vet3       :        Array [1..2] of Integer;     // que armazenam as ordens
14:      vet4       :        Array [1..2] of Integer;     // das matrizes
15:      vet5       :        Array [1..2] of Integer;
16:      vet6       :        Array [1..2] of Integer;
17:      vet7       :        Array [1..2] of Integer;
18:      aux1       :        Array [1..2] of Integer;     // Vetores auxiliares - auxiliam nas
    estruturas
19:      aux2       :        Array [1..2] of Integer;     // condicionais, copiando as ordens das
     matrizes.
20:      matA       :        Array [1..3,1..3] of Integer;
21:      matB       :        Array [1..3,1..4] of Integer;    // MatA a MatG - matrizes de preset
22:      matC       :        Array [1..3,1..1] of Integer;
23:      matD       :        Array [1..1,1..3] of Integer;
24:      matE       :        Array [1..3,1..4] of Integer;
25:      matF       :        Array [1..4,1..2] of Integer;
26:      matG       :        Array [1..3,1..3] of Integer;
27:      mop1       :        Array [1..5,1..5] of Integer;    // mop1 e mop2 - Auxiliam na
    realização de
28:      mop2       :        Array [1..5,1..5] of Integer;    // operações, copiando as matrizes
    de preset
29:      matS       :        Array [1..5,1..5] of Integer;
30:      matSub     :        Array [1..5,1..5] of Integer;    // MatS, MatSub, Matm2k e MatP são
    as
31:      matm2k     :        Array [1..5,1..5] of Integer;    // matrizes resultado das operações
32:      matP       :        Array [1..5,1..5] of Integer;
33:      continue   :        Char;                            // Continuar ou encerrar as
    atividades
34:      c          :        Integer;                         // Contador dos ciclos de escolha
    de operandos
35:      mop        :        Array [1..5,1..5] of Integer;    // Matriz operando geral para capta
    ção no case
36:      auxg       :        Array [1..2] of Integer;         // Vetor auxiliar geral para captaç
    ão no case
37:      ror1       :        Integer;                         // Variável resultante da
    multiplicação de matrizes [1,n] x [n,1]
38: begin
39:   //Declaração das matrizes de preset do exercício.
40:
41:   //Matriz A 3x3.
42:   matA[1,1] := 5;
43:   matA[1,2] := 2;
44:   matA[1,3] := 1;
45:   matA[2,1] := 15;
46:   matA[2,2] := 7;
47:   matA[2,3] := 7;
48:   matA[3,1] := 25;
49:   matA[3,2] := 7;
50:   matA[3,3] := 5;
51:
52:   //Matriz B 3x4.
53:   matB[1,1] := -5;
```

```
 54:   matB[1,2] := 2;
 55:   matB[1,3] := 1;
 56:   matB[1,4] := 1;
 57:   matB[2,1] := 15;
 58:   matB[2,2] := -7;
 59:   matB[2,3] := 7;
 60:   matB[2,4] := -1;
 61:   matB[3,1] := 25;
 62:   matB[3,2] := 7;
 63:   matB[3,3] := 5;
 64:   matB[3,4] := 3;
 65:
 66:   //Matriz C 3x1.
 67:   matC[1,1] := -2;
 68:   matC[2,1] := 1;
 69:   matC[3,1] := 4;
 70:
 71:   //Matriz D 1x3.
 72:   matD[1,1] := -2;
 73:   matD[1,2] := 0;
 74:   matD[1,3] := 5;
 75:
 76:   //Matriz E 3x4.
 77:   matE[1,1] := -5;
 78:   matE[1,2] := 2;
 79:   matE[1,3] := 1;
 80:   matE[1,4] := 1;
 81:   matE[2,1] := 15;
 82:   matE[2,2] := -7;
 83:   matE[2,3] := 7;
 84:   matE[2,4] := -1;
 85:   matE[3,1] := 25;
 86:   matE[3,2] := 7;
 87:   matE[3,3] := 5;
 88:   matE[3,4] := 3;
 89:
 90:   //Matriz F 4x2.
 91:   matF[1,1] := -2;
 92:   matF[1,2] := 0;
 93:   matF[2,1] := 1;
 94:   matF[2,2] := -1;
 95:   matF[3,1] := 5;
 96:   matF[3,2] := -4;
 97:   matF[4,1] := 12;
 98:   matF[4,2] := 3;
 99:
100:   //Matriz G 3x3.
101:   matG[1,1] := -2;
102:   matG[1,2] := -0;
103:   matG[1,3] := 2;
104:   matG[2,1] := 1;
105:   matG[2,2] := -1;
106:   matG[2,3] := 1;
107:   matG[3,1] := 5;
108:   matG[3,2] := -4;
109:   matG[3,3] := 0;
110:
111:   writeln('Bem vindo!');
112:   delay(1000);
113:   writeln('Este programa realiza calculos com matrizes predefinidas');
114:   writeln('Digite qualquer tecla para comecar');
115:   readkey;
116:   clrscr;                                  //Limpa a tela
```

```
117:   continue := 'y';
118:   delay(200);
119:   while continue <> 'n' do
120:   begin
121:       writeln('Exibindo as matrizes: ');
122:       writeln('Escolha as opcoes de 1 a 7 para os operandos conforme solicitado');
123:       writeln;
124:       // Matriz A.
125:       write('1 - A 3x3 =(');                          // Esses vetores e os semelhantes
     armazenam
126:       for i := 1 to 3 do                               // as ordens das matrizes para
     utilizar nas
127:           for j := 1 to 3 do                           // estruturas condicionais das operaçõ
     es de
128:               write(' ',matA[i,j],' ');                // adição, subtração e multiplicação
     entre
129:       write(')');                                      // matrizes.
130:       vet1[1] := i;
131:       vet1[2] := j;
132:       writeln;
133:       // Matriz B.
134:       write('2 - B 3x4 =(');
135:       for i := 1 to 3 do
136:           for j := 1 to 4 do
137:               write(' ',matB[i,j],' ');
138:       write(')');
139:       vet2[1] := i;
140:       vet2[2] := j;
141:       writeln;
142:       // Matriz C.
143:       write('3 - C 3x1 =(');
144:       for i := 1 to 3 do
145:           for j := 1 to 1 do
146:               write(' ',matC[i,j],' ');
147:       write(')');
148:       vet3[1] := i;
149:       vet3[2] := j;
150:       writeln;
151:       // Matriz D.
152:       write('4 - D 1x3 =(');
153:       for i := 1 to 1 do
154:           for j := 1 to 3 do
155:               write(' ',matD[i,j],' ');
156:       write(')');
157:       vet4[1] := i;
158:       vet4[2] := j;
159:       writeln;
160:       // Matriz E.
161:       write('5 - E 3x4 =(');
162:       for i := 1 to 3 do
163:           for j := 1 to 4 do
164:               write(' ',matE[i,j],' ');
165:       write(')');
166:       vet5[1] := i;
167:       vet5[2] := j;
168:       writeln;
169:       // Matriz F.
170:       write('6 - F 4x2 =(');
171:       for i := 1 to 4 do
172:           for j := 1 to 2 do
173:               write(' ',matF[i,j],' ');
174:       write(')');
175:       vet6[1] := i;
```

```
176:       vet6[2] := j;
177:       writeln;
178:       // Matriz G.
179:       write('7 - G 3x3 =(');
180:       for i := 1 to 3 do
181:           for j := 1 to 3 do
182:               write(' ',matG[i,j],' ');
183:       write(')');
184:       vet7[1] := i;
185:       vet7[2] := j;
186:       writeln;
187:       writeln;
188:       writeln('Escolha a operacao, ou 5 para sair: ');
189:       writeln('1 - Adicao de matrizes');
190:       writeln('2 - Subtracao de matrizes');
191:       writeln('3 - Multiplicacao por valor constante');
192:       writeln('4 - Multiplicacao de matrizes');
193:       writeln('5 - Sair');
194:       writeln;
195:       write('Digite a opcao: ');
196:       read(operacao);
197:       writeln('Pressione qualquer tecla para continuar');
198:       readkey;                //checkpoint
199:       case operacao of
200:            1 : begin                                  // Aqui começa a adição de matrizes
201:                  writeln;
202:                  writeln('Adicao de matrizes');
203:                  writeln;
204:                  for c := 1 to 2 do
205:                      begin
206:                         write('Escolha a ',c,'a matriz: ');          // Laço for para 1° e 2°
    operandos
207:                         read(op);
208:                         case op of
209:                              1 : begin       //Matriz A 3x3
210:                                    for i := 1 to 3 do
211:                                        for j := 1 to 3 do
212:                                            mop[i,j] := matA[i,j];
213:                                    write('A [',i,'x',j,'] = (');
214:                                    for i := 1 to 3 do
215:                                        for j := 1 to 3 do
216:                                            write(' ',mop[i,j],' ');
217:                                    write(')');
218:                                    auxg[1] := vet1[1];
219:                                    auxg[2] := vet1[2];
220:                                    writeln;
221:                                    writeln;
222:                                end;
223:                              2 : begin       //Matriz B 3x4
224:                                    for i := 1 to 3 do
225:                                        for j := 1 to 4 do
226:                                            mop[i,j] := matB[i,j];
227:                                    write('B [',i,'x',j,'] = (');
228:                                    for i := 1 to 3 do
229:                                        for j := 1 to 4 do
230:                                            write(' ',mop[i,j],' ');
231:                                    write(')');
232:                                    auxg[1] := vet2[1];
233:                                    auxg[2] := vet2[2];
234:                                    writeln;
235:                                    writeln;
236:                                end;
237:                              3 : begin       //Matriz C 3x1
```

```
238:                         for i := 1 to 3 do
239:                             for j := 1 to 1 do
240:                                 mop[i,j] := matC[i,j];
241:                         write('C [',i,'x',j,'] = (');
242:                         for i := 1 to 3 do
243:                             for j := 1 to 1 do
244:                                 write(' ',mop[i,j],' ');
245:                         write(')');
246:                         auxg[1] := vet3[1];
247:                         auxg[2] := vet3[2];
248:                         writeln;
249:                         writeln;
250:                       end;
251:                     4 : begin       //Matriz D 1x3
252:                         for i := 1 to 1 do
253:                             for j := 1 to 3 do
254:                                 mop[i,j] := matD[i,j];
255:                         write('D [',i,'x',j,'] = (');
256:                         for i := 1 to 1 do
257:                             for j := 1 to 3 do
258:                                 write(' ',mop[i,j],' ');
259:                         write(')');
260:                         auxg[1] := vet4[1];
261:                         auxg[2] := vet4[2];
262:                         writeln;
263:                         writeln;
264:                       end;
265:                     5 : begin       //Matriz E 3x4
266:                         for i := 1 to 3 do
267:                             for j := 1 to 4 do
268:                                 mop[i,j] := matE[i,j];
269:                         write('E [',i,'x',j,'] = (');
270:                         for i := 1 to 3 do
271:                             for j := 1 to 4 do
272:                                 write(' ',mop[i,j],' ');
273:                         write(')');
274:                         auxg[1] := vet5[1];
275:                         auxg[2] := vet5[2];
276:                         writeln;
277:                         writeln;
278:                       end;
279:                     6 : begin       //Matriz F 4x2
280:                         for i := 1 to 4 do
281:                             for j := 1 to 2 do
282:                                 mop[i,j] := matF[i,j];
283:                         write('F [',i,'x',j,'] = (');
284:                         for i := 1 to 4 do
285:                             for j := 1 to 2 do
286:                                 write(' ',mop[i,j],' ');
287:                         write(')');
288:                         auxg[1] := vet6[1];
289:                         auxg[2] := vet6[2];
290:                         writeln;
291:                         writeln;
292:                       end;
293:                     7 : begin       //Matriz G 3x3
294:                         for i := 1 to 3 do
295:                             for j := 1 to 3 do
296:                                 mop[i,j] := matG[i,j];
297:                         write('G [',i,'x',j,'] = (');
298:                         for i := 1 to 3 do
299:                             for j := 1 to 3 do
300:                                 write(' ',mop[i,j],' ');
```

```
301:                                   write(')');
302:                                   auxg[1] := vet7[1];
303:                                   auxg[2] := vet7[2];
304:                                   writeln;
305:                                   writeln;
306:                                 end
307:                             else
308:                                 begin
309:                                   writeln('Opcao invalida!');
310:                                   readkey;
311:                                   clrscr;
312:                                 end;                                        //Os laços for
    perfazem os ciclos de leitura e gravação.
313:                         end;                                                //O contador c é
    usado nas estruturas if para separar o que é do operando 1 e o que é do 2.
314:                         if c = 1 then                                       //Nas estruturas
    case, foram utilizadas variáveis auxiliares para
315:                         begin                                               // atuarem de
    modo geral nas contagens e gravações.
316:                             for i := 1 to auxg[1] do                        // São elas:
317:                                 for j := 1 to auxg[2] do                            //mop:
    matriz auxiliar geral.
318:                                     mop1[i,j] := mop[i,j];                  //auxg:
    vetor auxiliar geral.
319:                             aux1[1] := auxg[1];
320:                             aux1[2] := auxg[2];
321:                         end;
322:                         if c = 2 then                                       //Nas estruturas
    if, para c em cada ciclo, essas variáveis gerais
323:                         begin                                               //são copiadas
    para as variáveis auxiliares específicas de cada ciclo.
324:                             for i := 1 to auxg[1] do                        //No ciclo 1 (c =
     1), as variáveis receptoras são as especícicas do
325:                                 for j := 1 to auxg[2] do                    // 1° operando!
        mop1 [i,j] e aux1
326:                                     mop2[i,j] := mop[i,j];                  //No ciclo 2 (c =
     2), as variáveis receptoras são as específicas do
327:                             aux2[1] := auxg[1];                             // 2° operando!
        mop2 [i,j] e aux2
328:                             aux2[2] := auxg[2];
329:                         end;
330:                     end;
331:                     if(aux1[1] = aux2[1]) and (aux1[2] = aux2[2]) then     // Estrutura
    condicional
332:                         begin
333:                         writeln('Adicao de matrizes:');                     // Operação de
    adição
334:                         writeln;
335:                         for i := 1 to aux1[1] do
336:                             for j := 1 to aux1[2] do
337:                                 matS[i,j] := mop1[i,j] + mop2[i,j];
338:                         write('Matriz soma [',i,'x',j,']',' = ','(');
339:                         for i := 1 to aux1[1] do
340:                             for j := 1 to aux2[1] do
341:                                 write(' ',matS[i,j],' ');
342:                         write(')');
343:                         writeln;
344:                         writeln;
345:                         writeln('Fim da operacao');
346:                         readkey;
347:                         clrscr;
348:                         end
349:                     else
```

```
350:                         begin
351:                           writeln('Operacao impossivel devido a ordem das matrizes');
352:                           writeln('Deseja continuar? (y/n)');
353:                           continue := readkey;
354:                           clrscr;
355:                         end;
356:                end;
357:             2 : begin
358:                   writeln('Subtracao de matrizes');
359:                   writeln;
360:                   for c := 1 to 2 do
361:                       begin
362:                          write('Escolha a ',c,'a matriz: ');          // Laço for para 1° e 2°
    operandos
363:                          read(op);
364:                          case op of
365:                               1 : begin       //Matriz A 3x3
366:                                     for i := 1 to 3 do
367:                                         for j := 1 to 3 do
368:                                             mop[i,j] := matA[i,j];
369:                                     write('A [',i,'x',j,'] = (');
370:                                     for i := 1 to 3 do
371:                                         for j := 1 to 3 do
372:                                             write(' ',mop[i,j],' ');
373:                                     write(')');
374:                                     auxg[1] := vet1[1];
375:                                     auxg[2] := vet1[2];
376:                                     writeln;
377:                                     writeln;
378:                                   end;
379:                               2 : begin       //Matriz B 3x4
380:                                     for i := 1 to 3 do
381:                                         for j := 1 to 4 do
382:                                             mop[i,j] := matB[i,j];
383:                                     write('B [',i,'x',j,'] = (');
384:                                     for i := 1 to 3 do
385:                                         for j := 1 to 4 do
386:                                             write(' ',mop[i,j],' ');
387:                                     write(')');
388:                                     auxg[1] := vet2[1];
389:                                     auxg[2] := vet2[2];
390:                                     writeln;
391:                                     writeln;
392:                                   end;
393:                               3 : begin       //Matriz C 3x1
394:                                     for i := 1 to 3 do
395:                                         for j := 1 to 1 do
396:                                             mop[i,j] := matC[i,j];
397:                                     write('B [',i,'x',j,'] = (');
398:                                     for i := 1 to 3 do
399:                                         for j := 1 to 1 do
400:                                             write(' ',mop[i,j],' ');
401:                                     write(')');
402:                                     auxg[1] := vet3[1];
403:                                     auxg[2] := vet3[2];
404:                                     writeln;
405:                                     writeln;
406:                                   end;
407:                               4 : begin       //Matriz D 1x3
408:                                     for i := 1 to 1 do
409:                                         for j := 1 to 3 do
410:                                             mop[i,j] := matD[i,j];
411:                                     write('D [',i,'x',j,'] = (');
```

```
412:                             for i := 1 to 1 do
413:                                 for j := 1 to 3 do
414:                                     write(' ',mop[i,j],' ');
415:                             write(')');
416:                             auxg[1] := vet4[1];
417:                             auxg[2] := vet4[2];
418:                             writeln;
419:                             writeln;
420:                         end;
421:                     5 : begin       //Matriz E 3x4
422:                             for i := 1 to 3 do
423:                                 for j := 1 to 4 do
424:                                     mop[i,j] := matE[i,j];
425:                             write('E [',i,'x',j,'] = (');
426:                             for i := 1 to 3 do
427:                                 for j := 1 to 4 do
428:                                     write(' ',mop[i,j],' ');
429:                             write(')');
430:                             auxg[1] := vet5[1];
431:                             auxg[2] := vet5[2];
432:                             writeln;
433:                             writeln;
434:                         end;
435:                     6 : begin       //Matriz F 4x2
436:                             for i := 1 to 4 do
437:                                 for j := 1 to 2 do
438:                                     mop[i,j] := matF[i,j];
439:                             write('F [',i,'x',j,'] = (');
440:                             for i := 1 to 4 do
441:                                 for j := 1 to 2 do
442:                                     write(' ',mop[i,j],' ');
443:                             write(')');
444:                             auxg[1] := vet6[1];
445:                             auxg[2] := vet6[2];
446:                             writeln;
447:                             writeln;
448:                         end;
449:                     7 : begin       //Matriz G 3x3
450:                             for i := 1 to 3 do
451:                                 for j := 1 to 3 do
452:                                     mop[i,j] := matG[i,j];
453:                             write('G [',i,'x',j,'] = (');
454:                             for i := 1 to 3 do
455:                                 for j := 1 to 3 do
456:                                     write(' ',mop[i,j],' ');
457:                             write(')');
458:                             auxg[1] := vet7[1];
459:                             auxg[2] := vet7[2];
460:                             writeln;
461:                             writeln;
462:                         end
463:                     else
464:                         begin
465:                             writeln('Opcao invalida!');
466:                             readkey;
467:                             clrscr;
468:                         end;
469:                 end;
470:                 if c = 1 then
471:                 begin
472:                     for i := 1 to auxg[1] do
473:                         for j := 1 to auxg[2] do
474:                             mop1[i,j] := mop[i,j];
```

```
475:                          aux1[1] := auxg[1];
476:                          aux1[2] := auxg[2];
477:                       end;
478:                       if c = 2 then
479:                       begin
480:                          for i := 1 to auxg[1] do
481:                              for j := 1 to auxg[2] do
482:                                  mop2[i,j] := mop[i,j];
483:                          aux2[1] := auxg[1];
484:                          aux2[2] := auxg[2];
485:                       end;
486:                    end;
487:                    if(aux1[1] = aux2[1]) and (aux1[2] = aux2[2]) then     // Estrutura
    condicional
488:                       begin
489:                       writeln('Subtracao de matrizes:');                      // Operação
    de subtração
490:                       writeln;
491:                       for i := 1 to aux1[1] do
492:                           for j := 1 to aux1[2] do
493:                               matSub[i,j] := mop1[i,j] - mop2[i,j];
494:                       write('Matriz diferenca [',i,'x',j,']',' = ','(');
495:                       for i := 1 to aux1[1] do
496:                           for j := 1 to aux2[1] do
497:                               write(' ',matSub[i,j],' ');
498:                       write(')');
499:                       writeln;
500:                       writeln;
501:                       writeln('Fim da operacao');
502:                       readkey;
503:                       clrscr;
504:                       end
505:                    else
506:                       begin
507:                         writeln('Operacao impossivel devido a ordem das matrizes');
508:                         writeln('Deseja continuar? (y/n)');
509:                         continue := readkey;
510:                         clrscr;
511:                       end;
512:              end;
513:           3 : begin                                  //Multiplicação de matriz por valor real
514:                 writeln('Multiplicacao por escalar');
515:                 writeln;
516:                 write('Escolha uma matriz: ');
517:                 read(op);                                          // Matriz - escolha
518:                 case op of
519:                      1 : begin       //Matriz A 3x3
520:                            for i := 1 to 3 do
521:                                for j := 1 to 3 do
522:                                    mop[i,j] := matA[i,j];
523:                            write('A [',i,'x',j,'] = (');
524:                            for i := 1 to 3 do
525:                                for j := 1 to 3 do
526:                                    write(' ',mop[i,j],' ');
527:                            write(')');
528:                            auxg[1] := vet1[1];
529:                            auxg[2] := vet1[2];
530:                            writeln;
531:                            writeln;
532:                          end;
533:                      2 : begin       //Matriz B 3x4
534:                            for i := 1 to 3 do
535:                                for j := 1 to 4 do
```

```
536:                             mop[i,j] := matB[i,j];
537:                       write('B [',i,'x',j,'] = (');
538:                       for i := 1 to 3 do
539:                           for j := 1 to 4 do
540:                               write(' ',mop[i,j],' ');
541:                       write(')');
542:                       auxg[1] := vet2[1];
543:                       auxg[2] := vet2[2];
544:                       writeln;
545:                       writeln;
546:                   end;
547:               3 : begin       //Matriz C 3x1
548:                       for i := 1 to 3 do
549:                           for j := 1 to 1 do
550:                               mop[i,j] := matC[i,j];
551:                       write('C [',i,'x',j,'] = (');
552:                       for i := 1 to 3 do
553:                           for j := 1 to 1 do
554:                               write(' ',mop[i,j],' ');
555:                       write(')');
556:                       auxg[1] := vet3[1];
557:                       auxg[2] := vet3[2];
558:                       writeln;
559:                       writeln;
560:                   end;
561:               4 : begin       //Matriz D 1x3
562:                       for i := 1 to 1 do
563:                           for j := 1 to 3 do
564:                               mop[i,j] := matD[i,j];
565:                       write('D [',i,'x',j,'] = (');
566:                       for i := 1 to 1 do
567:                           for j := 1 to 3 do
568:                               write(' ',mop[i,j],' ');
569:                       write(')');
570:                       auxg[1] := vet4[1];
571:                       auxg[2] := vet4[2];
572:                       writeln;
573:                       writeln;
574:                   end;
575:               5 : begin       //Matriz E 3x4
576:                       for i := 1 to 3 do
577:                           for j := 1 to 4 do
578:                               mop[i,j] := matE[i,j];
579:                       write('E [',i,'x',j,'] = (');
580:                       for i := 1 to 3 do
581:                           for j := 1 to 4 do
582:                               write(' ',mop[i,j],' ');
583:                       write(')');
584:                       auxg[1] := vet5[1];
585:                       auxg[2] := vet5[2];
586:                       writeln;
587:                       writeln;
588:                   end;
589:               6 : begin       //Matriz F 4x2
590:                       for i := 1 to 4 do
591:                           for j := 1 to 2 do
592:                               mop[i,j] := matF[i,j];
593:                       write('F [',i,'x',j,'] = (');
594:                       for i := 1 to 4 do
595:                           for j := 1 to 2 do
596:                               write(' ',mop[i,j],' ');
597:                       write(')');
598:                       auxg[1] := vet6[1];
```

```
599:                           auxg[2] := vet6[2];
600:                           writeln;
601:                           writeln;
602:                         end;
603:                     7 : begin       //Matriz G 3x3
604:                           for i := 1 to 3 do
605:                               for j := 1 to 3 do
606:                                   mop[i,j] := matG[i,j];
607:                           write('G [',i,'x',j,'] = (');
608:                           for i := 1 to 3 do
609:                               for j := 1 to 3 do
610:                                   write(' ',mop[i,j],' ');
611:                           write(')');
612:                           auxg[1] := vet7[1];
613:                           auxg[2] := vet7[2];
614:                           writeln;
615:                           writeln;
616:                         end
617:                     else
618:                         begin
619:                             writeln('Opcao invalida!');
620:                             readkey;
621:                             clrscr;
622:                         end;
623:                 end;
624:                 write('Digite o valor real: ');                 // Número real
625:                 read(vr);
626:                 writeln;
627:                 writeln('Multiplicacao de matrizes por escalar:');    // Multiplicação por n
    úmero real
628:                 writeln;
629:                 for i := 1 to auxg[1] do
630:                     for j := 1 to auxg[2] do
631:                         matm2k[i,j] := mop[i,j] * vr;
632:                 write('Matriz Produto por escalar [',i,'x',j,']',' = ','(');
633:                 for i := 1 to auxg[1] do
634:                     for j := 1 to auxg[2] do
635:                         write(' ',matm2k[i,j],' ');
636:                 write(')');
637:                 writeln;
638:                 writeln;
639:                 writeln('Fim da operacao');
640:                 readkey;
641:                 clrscr;
642:               end;                                              // Fim
643:           4 : begin
644:                 writeln('Multiplicacao de matrizes');
645:                 writeln;
646:                 for c := 1 to 2 do
647:                     begin
648:                        write('Escolha a ',c,'a matriz: ');          // Laço for para 1° e 2°
     operandos
649:                        read(op);
650:                        case op of
651:                             1 : begin       //Matriz A 3x3
652:                                   for i := 1 to 3 do
653:                                       for j := 1 to 3 do
654:                                           mop[i,j] := matA[i,j];
655:                                   write('A [',i,'x',j,'] = (');
656:                                   for i := 1 to 3 do
657:                                       for j := 1 to 3 do
658:                                           write(' ',mop[i,j],' ');
659:                                   write(')');
```

```
660:                                auxg[1] := vet1[1];
661:                                auxg[2] := vet1[2];
662:                                writeln;
663:                                writeln;
664:                            end;
665:                        2 : begin       //Matriz B 3x4
666:                                for i := 1 to 3 do
667:                                    for j := 1 to 4 do
668:                                        mop[i,j] := matB[i,j];
669:                                write('B [',i,'x',j,'] = (');
670:                                for i := 1 to 3 do
671:                                    for j := 1 to 4 do
672:                                        write(' ',mop[i,j],' ');
673:                                write(')');
674:                                auxg[1] := vet2[1];
675:                                auxg[2] := vet2[2];
676:                                writeln;
677:                                writeln;
678:                            end;
679:                        3 : begin       //Matriz C 3x1
680:                                for i := 1 to 3 do
681:                                    for j := 1 to 1 do
682:                                        mop[i,j] := matC[i,j];
683:                                write('C [',i,'x',j,'] = (');
684:                                for i := 1 to 3 do
685:                                    for j := 1 to 1 do
686:                                        write(' ',mop[i,j],' ');
687:                                write(')');
688:                                auxg[1] := vet3[1];
689:                                auxg[2] := vet3[2];
690:                                writeln;
691:                                writeln;
692:                            end;
693:                        4 : begin       //Matriz D 1x3
694:                                for i := 1 to 1 do
695:                                    for j := 1 to 3 do
696:                                        mop[i,j] := matD[i,j];
697:                                write('D [',i,'x',j,'] = (');
698:                                for i := 1 to 1 do
699:                                    for j := 1 to 3 do
700:                                        write(' ',mop[i,j],' ');
701:                                write(')');
702:                                auxg[1] := vet4[1];
703:                                auxg[2] := vet4[2];
704:                                writeln;
705:                                writeln;
706:                            end;
707:                        5 : begin       //Matriz E 3x4
708:                                for i := 1 to 3 do
709:                                    for j := 1 to 4 do
710:                                        mop[i,j] := matE[i,j];
711:                                write('E [',i,'x',j,'] = (');
712:                                for i := 1 to 3 do
713:                                    for j := 1 to 4 do
714:                                        write(' ',mop[i,j],' ');
715:                                write(')');
716:                                auxg[1] := vet5[1];
717:                                auxg[2] := vet5[2];
718:                                writeln;
719:                                writeln;
720:                            end;
721:                        6 : begin       //Matriz F 4x2
722:                                for i := 1 to 4 do
```

```
                                    for j := 1 to 2 do
                                        mop[i,j] := matF[i,j];
                                  write('F [',i,'x',j,'] = (');
                                  for i := 1 to 4 do
                                      for j := 1 to 2 do
                                          write(' ',mop[i,j],' ');
                                  write(')');
                                  auxg[1] := vet6[1];
                                  auxg[2] := vet6[2];
                                  writeln;
                                  writeln;
                                end;
                           7 : begin       //Matriz G 3x3
                                  for i := 1 to 3 do
                                      for j := 1 to 3 do
                                          mop[i,j] := matG[i,j];
                                  write('G [',i,'x',j,'] = (');
                                  for i := 1 to 3 do
                                      for j := 1 to 3 do
                                          write(' ',mop[i,j],' ');
                                  write(')');
                                  auxg[1] := vet7[1];
                                  auxg[2] := vet7[2];
                                  writeln;
                                  writeln;
                                end
                           else
                               begin
                                  writeln('Opcao invalida!');
                                  readkey;
                                  clrscr;
                                end;
                        end;
                        if c = 1 then
                        begin
                             for i := 1 to auxg[1] do
                                 for j := 1 to auxg[2] do
                                     mop1[i,j] := mop[i,j];
                             aux1[1] := auxg[1];
                             aux1[2] := auxg[2];
                        end;
                        if c = 2 then
                        begin
                             for i := 1 to auxg[1] do
                                 for j := 1 to auxg[2] do
                                     mop2[i,j] := mop[i,j];
                             aux2[1] := auxg[1];
                             aux2[2] := auxg[2];
                        end;
                    end;
                    readkey;
                    if(aux1[2] = aux2[1]) then              // Estrutura condicional
                       begin
                       for i := 1 to 5 do
                           for j := 1 to 5 do                    // Zerando a matriz
                               matP[i,j] := 0;                   // produto - matP
                       if (aux1[1] = 1) and (aux2[2] = 1) then
                       begin
                         writeln('Multiplicacao de matrizes:');        //Matriz linha por
    matriz coluna
                         writeln;
                         for i := 1 to aux1[2] do
     // Para este caso, foi necessário forçar uma mudança na ordem das matrizes
```

```
                              for j := 1 to aux2[1] do
    // para poder aproveitar a mesma função de multiplicação, pois esta não atualizava
                                 for k := 1 to aux1[2] do
    // seus valores para matrizes produto matP[1x1]. Para isso, usamos os números de
                                    matP[i,j] := mop1[i,k] * mop2[k,j] + matP[i,j];
    // de coluna da primeira matriz e de linha da segunda, para forçar as contagens dos
                           ror1 := matP[1,1];
    // laços i,j e k, e assim, forçar a continuidade da função no decorrer dos ciclos.
                           writeln('Matriz produto [',1,'x',1,'] = (',ror1,')');
    // Veja que apareceram vários zeros na amostragem, então o verdadeiro valor da matriz
                           writeln;
    // , em [1,1], foi copiado para uma variável resultado integer para uma amostragem única.
                           writeln('Fim da operacao');
                           readkey;
                           clrscr;
                         end
                         else
                           begin
                             writeln('Multiplicacao de matrizes:');     // Operação de
    multiplicação M2M padrão
                             for i := 1 to aux1[1] do
                                 for j := 1 to aux2[2] do
                                     for k := 1 to aux1[1] do
                                         matP[i,j] := mop1[i,k] * mop2[k,j] + matP[i,j];
                             write('Matriz Produto [',i,'x',j,']',' = ','(');
                             for i := 1 to aux1[1] do
                                 for j := 1 to aux2[2] do
                                     write(' ',matP[i,j],' ');
                             write(')');
                             writeln;
                             writeln;
                             writeln('Fim da operacao');
                             readkey;
                             clrscr;
                           end
                       end
                     else
                       begin
                         writeln('Impossivel, pois o numero de colunas da primeira matriz');
                         writeln('nao eh igual ao numero de linhas da segunda matriz!');
                         writeln('Deseja continuar? (y/n)');
                         continue := readkey;
                         clrscr;
                       end;
                end;                                              // Fim - M2M
            5 : begin                                             // Sair ou continuar
                  clrscr;
                  writeln('Sair');
                  writeln('Deseja continuar? (y/n)');
                  continue := readkey;
                  clrscr;
                end
            else
                begin
                  writeln ('Operacao invalida!!');
                  readkey;
                  clrscr;
                end;
        end;
    end;
end.
```