*Analisis*

**Entrada:** Solo los limites de las matrices, todas las operaciones las presento secuencialmente.

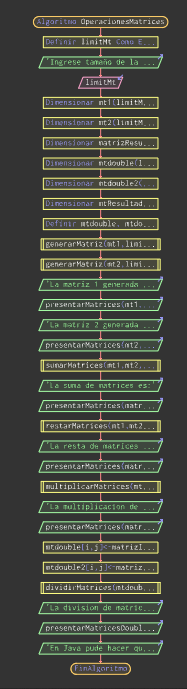
**Proceso:** Para poder hacer los cálculos use procedimientos, en donde un procedimiento es solo la versión sobrecargada de el mismo procedimiento (sumarMatrices), y los presento a todos a través de una función que retorna valor en cadena (presentarMatrices), todos ellos son invocados desde el procedimiento principal o método main en java. Creo adicionalmente otros métodos/funciones para poder convertir las matrices de enteros a tipo decimal y asi poder ejercer división entre ellos.

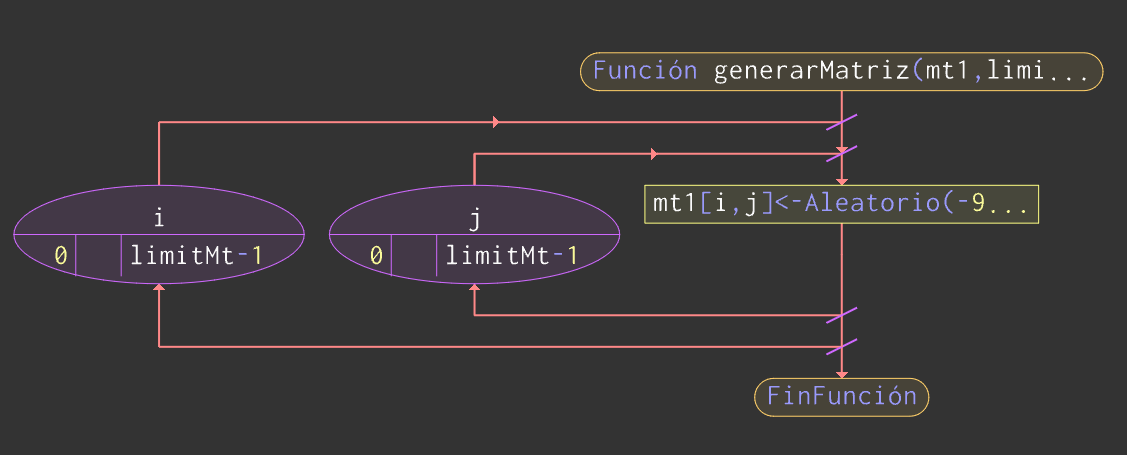
**Salida:** La suma, resta o multiplicación de las matrices, y en el caso de la división: la división entre el equivalente Real de las matrices 1 y 2 generadas

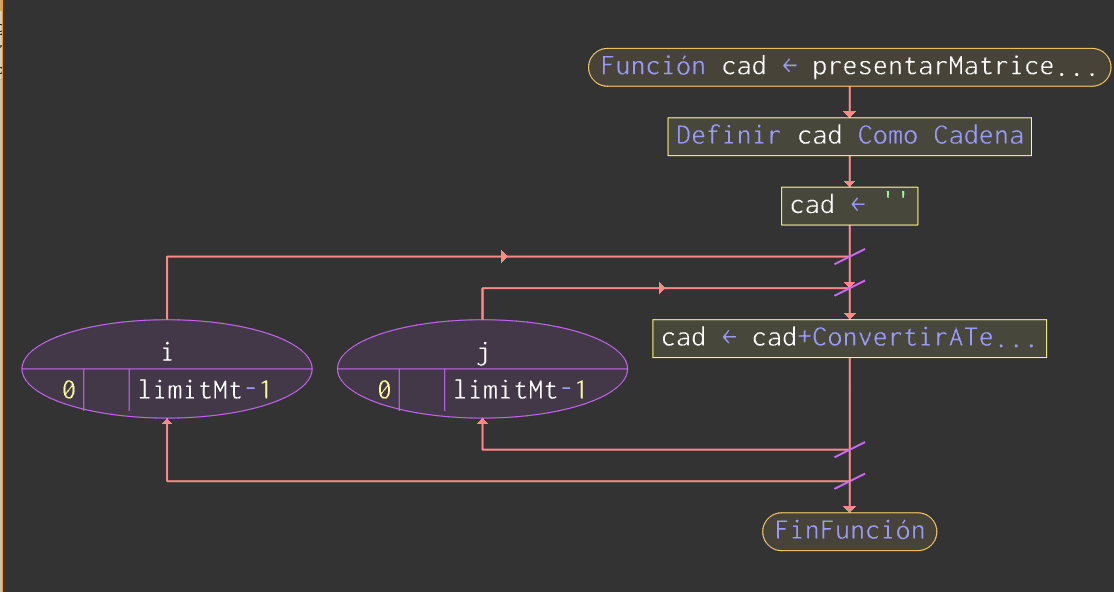
*Pseudocodigo:*

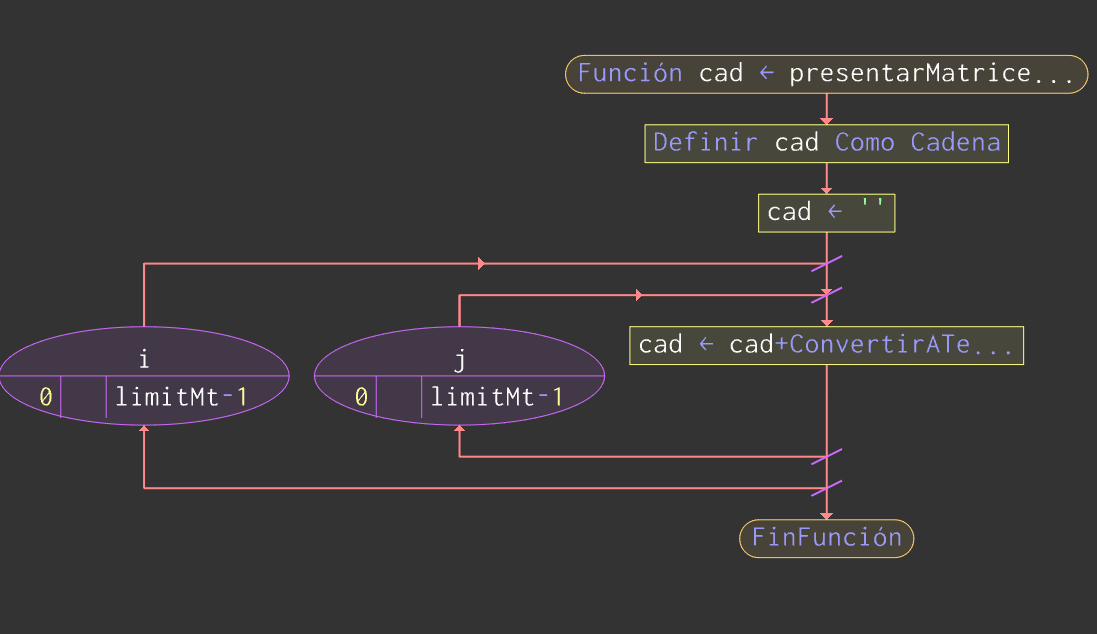
1. Algoritmo OperacionesMatrices
2. Definir limitMt Como Entero
3. Escribir "Ingrese tamaño de la matriz"
4. Leer limitMt
5. Dimension mt1[limitMt,limitMt]
6. Dimension mt2[limitMt,limitMt]
7. Dimension matrizResultado[limitMt,limitMt]
8. Dimension mtdouble[limitMt,limitMt]
9. Dimension mtdouble2[limitMt,limitMt]
10. Dimension mtResultadodouble[limitMt,limitMt]
11. generarMatriz(mt1,limitMt)
12. generarMatriz(mt2,limitMt)
13. Escribir "La matriz 1 generada es:"
14. Escribir presentarMatrices(mt1,limitMt)
15. Escribir "La matriz 2 generada es:"
16. Escribir presentarMatrices(mt2,limitMt)
17. sumarMatrices(mt1,mt2,matrizResultado,limitMt)
18. Escribir "La suma de matrices es:"
19. Escribir presentarMatrices(matrizResultado,limitMt)
20. restarMatrices(mt1,mt2,matrizResultado,limitMt)
21. Escribir "La resta de matrices es:"
22. Escribir presentarMatrices(matrizResultado,limitMt)
23. multiplicarMatrices(mt1,mt2,matrizResultado,limitMt)
24. Escribir "La multiplicacion de matrices es:"
25. Escribir presentarMatrices(matrizResultado,limitMt)
26. mtdouble[i,j] = matrizInt\_a\_MatrizDouble(mt1, mtdouble, limitMt)
27. mtdouble2[i,j] = matrizInt\_a\_MatrizDouble(mt2,mtdouble, limitMt)
28. dividirMatrices(mtdouble,mtdouble2,mtResultadodouble,limitMt)
29. Escribir "La division de matrices es:"
30. Escribir presentarMatricesDouble(mtResultadodouble,limitMt)
31. Escribir "En Java pude hacer que las divisiones no fueran 0"
32. FinAlgoritmo
33. SubProceso generarMatriz(mt1,limitMt)
34. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. mt1[i,j] <- Aleatorio(-9,10)
      2. FinPara
35. FinPara
36. FinSubProceso
37. Funcion cad = presentarMatrices(mt1,limitMt)
38. Definir cad Como Cadena
39. cad <- ""
40. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. cad <- cad + ConvertirATexto(mt1[i,j]) + " " + Tabulador
      2. FinPara
41. FinPara
42. FinFuncion
43. Funcion cad = presentarMatricesDouble(mtdouble,limitMt)
44. Definir cad Como Cadena
45. cad <- ""
46. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. cad <- cad + ConvertirATexto(mtdouble[i,j])+ " " + Tabulador
      2. FinPara
47. FinPara
48. FinFuncion
49. SubProceso sumarMatrices(mt1, mt2, matrizResultado,limitMt)
50. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. matrizResultado[i,j] <- mt1[i,j] + mt2[i,j]
      2. FinPara
51. FinPara
52. FinSubProceso
53. SubProceso restarMatrices(mt1, mt2, matrizResultado,limitMt)
54. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. matrizResultado[i,j] <- mt1[i,j] - mt2[i,j]
      2. FinPara
55. FinPara
56. FinSubProceso
57. SubProceso multiplicarMatrices(mt1,mt2, matrizResultado,limitMt)
58. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         + 1. matrizResultado[i,j] <- mt1[i,j] \* mt2[i,j]
      2. FinPara
59. FinPara
60. FinSubProceso
61. Funcion mtdouble = matrizInt\_a\_MatrizDouble(mt1, mtdouble, limitMt)
62. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. mtdouble[i,j] <- mt1[i,j] // ConvertirAReal
      2. FinPara
63. FinPara
64. FinSubProceso
65. SubProceso dividirMatrices(mtdouble, mtdouble2, mtResultadodouble,limitMt)
66. Para i<-0 Hasta limitMt-1 Hacer
    * 1. Para j<-0 Hasta limitMt-1 Hacer
         1. Si (mtdouble[i,j]<>0 Y mtResultadodouble[i,j]<>0 Y mtdouble2<>0)
            1. mtResultadodouble[i,j] = mtdouble / mtdouble2;
         2. FinSi
      2. FinPara
67. FinPara
68. FinSubProceso

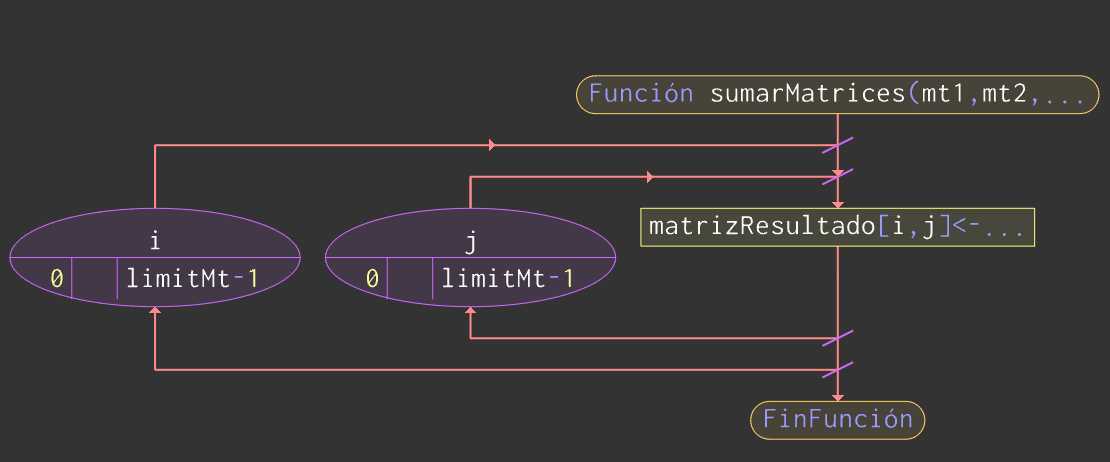
*Flujograma:*

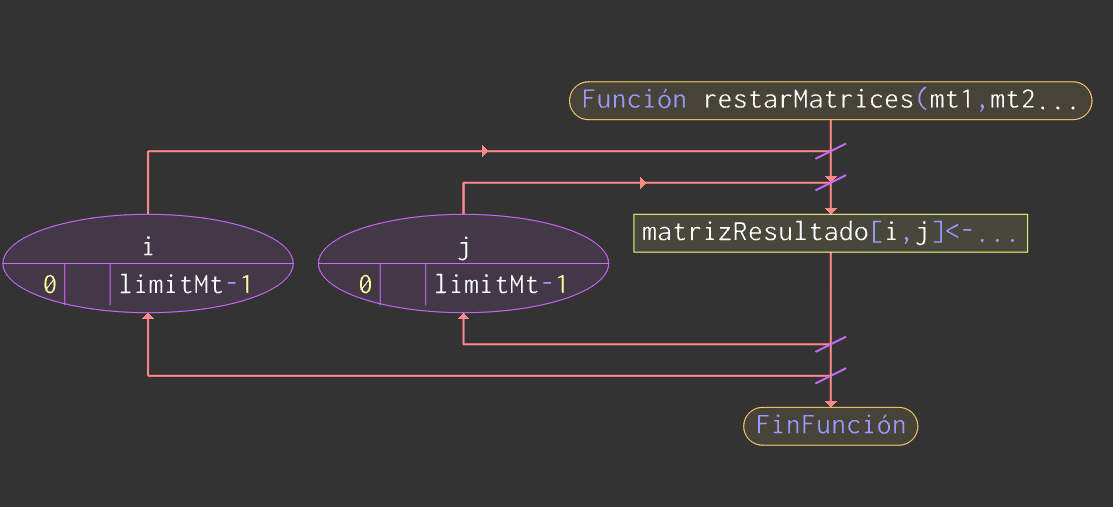


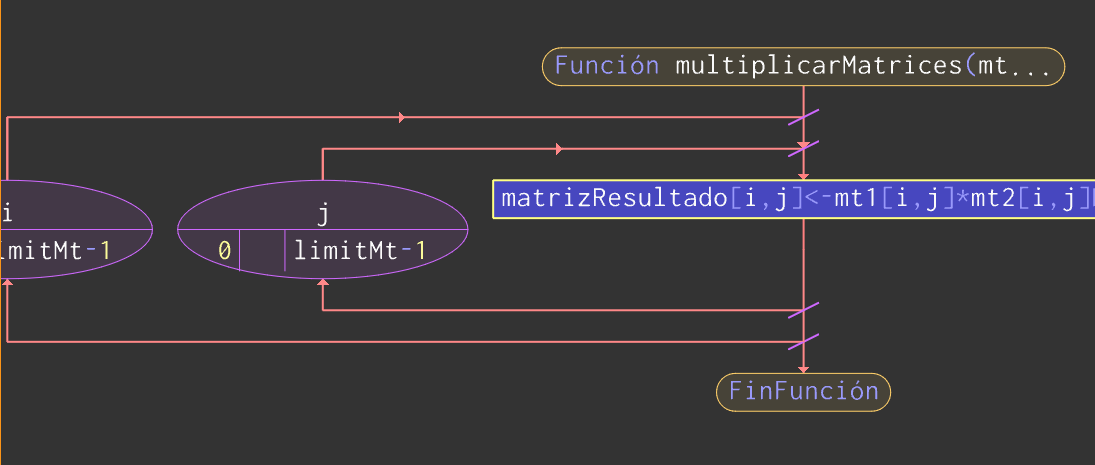


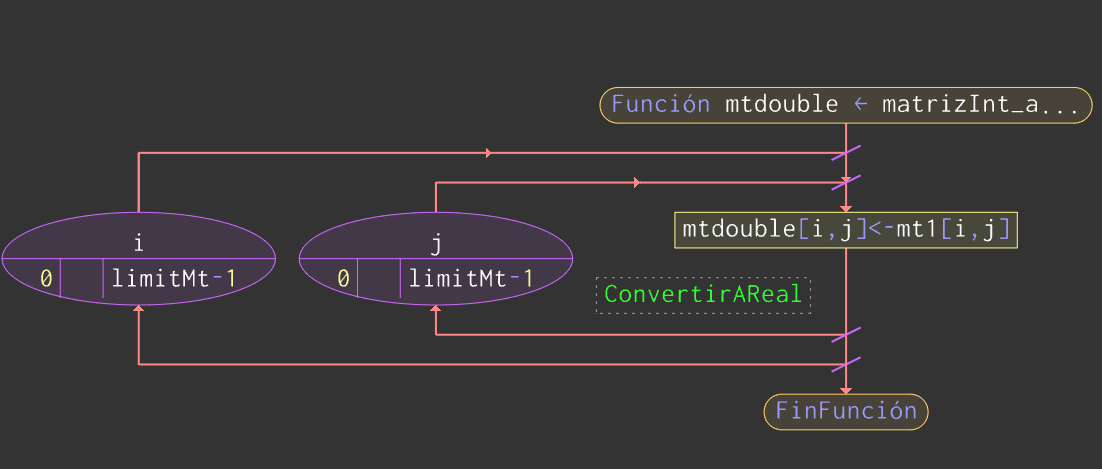


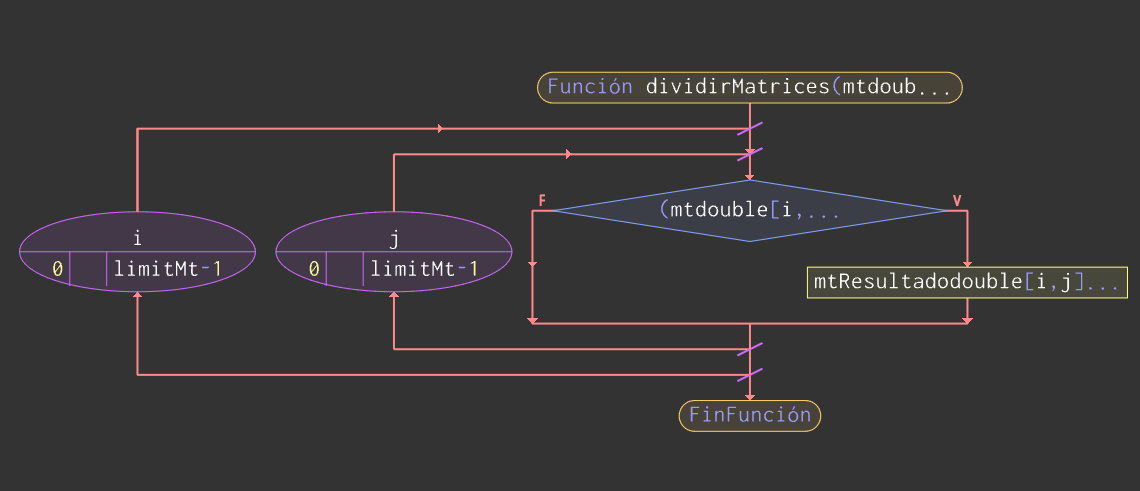












*Pruebas de Escritorio:*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nota: Cuando un numero de [i] tiene su valor de al lado vacio es porque avanza su posicion en [j] ( hice esto para evitar que sea largo el ejercicio como las anteriores pruebas de escritorio que he hecho) | | | | | | | | | |
| mt1 | | mt2 | | mtdouble | | mtdouble2 | | matrizResultado ( al multiplicar) | |
| [0] |  | [0] |  | [0] |  | [0] |  | [0] |  |
| [0] | 7 | [0] | 1 | [0] | 7 | [0] | 1 | [0] | 7 |
| [1] | -3 | [1] | 0 | [1] | -3 | [1] | 0 | [1] | 0 |
| [2] | 7 | [2] | -4 | [2] | 7 | [2] | -4 | [2] | -28 |
| [1] |  | [1] |  | [1] |  | [1] |  | [1] |  |
| [0] | 8 | [0] | -2 | [0] | 8 | [0] | -2 | [0] | -16 |
| [1] | -1 | [1] | 7 | [1] | -1 | [1] | 7 | [1] | -7 |
| [2] | -8 | [2] | -5 | [2] | -8 | [2] | -5 | [2] | 40 |
| [2] |  | [2] |  | [2] |  | [2] |  | [2] |  |
| [0] | -2 | [0] | -5 | [0] | -2 | [0] | -5 | [0] | 10 |
| [1] | 3 | [1] | 7 | [1] | 3 | [1] | 7 | [1] | 21 |
| [2] | -7 | [2] | 3 | [2] | -7 | [2] | 3 | [2] | -21 |