

Práctica 1

Calculadora de Matrices

Luis Eduardo Galindo Amaya (1274895)



Asignatura	Herramientas de Desarrollo de Software (40017)
Docente	Manuel Castañón-Puga
Fecha	2022-08-24 mié

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
Facultad de Ciencias Químicas e Ingeniería

Programa de Ingeniero en Software y Tecnologías Emergentes

1. Código A

[1-1]

2. Código B

```
1
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 /* #define MAX_MATRIX_SIZE 8 */
7
8 void inicializar_matriz(int[][8]);
9 int seleccionar_operacion();
10 void matriz_valida(size_t, size_t);
11 void mostrar_matriz(int[][8], size_t, size_t);
12 void captura_matriz(int[][8], size_t, size_t);
13 void sumar_matriz(int[][8], int[][8], int[][8]);
14 void multiplicar_matriz(int[][8], int[][8], int[][8], int, int, int, int);
15 void transpuesta_matriz(int[][8], int[][8]);
16
17 int main() {
18     /* matrices de entrada */
19     int A[8][8];
20     int B[8][8];
21     /* matriz resultado */
22     int R[8][8];
23
24     inicializar_matriz(A);
25     inicializar_matriz(B);
26     inicializar_matriz(R);
27
28     /* dimensiones de A */
29     int A_filas = 2;
30     int A_columnas = 3;
31
32     /* dimensiones de B */
33     int B_filas = 2;
```

```
34 int B_columnas = 3;
35
36 switch (seleccionar_operacion()) {
37 case 0:
38     printf("Adios :(");
39     break;
40
41 case 1:
42     /* La suma de matrices solo de puede hacer */
43     /* entre matrices cuadradas */
44
45     printf("Suma de matrices\n\n");
46     printf("dimensiones (CxR): ");
47     scanf("%dx%d", &A_filas, &A_columnas);
48
49     /* verificar que las matrices sean validas */
50     matriz_valida(A_filas, A_columnas);
51
52     printf("Matriz A:\n");
53     captura_matriz(A, A_filas, A_columnas);
54     printf("Matriz B:\n");
55     captura_matriz(B, A_filas, A_columnas);
56
57     sumar_matriz(A, B, R);
58
59     printf("\nMatriz A:\n");
60     mostrar_matriz(A, A_filas, A_columnas);
61     printf("\nMatriz B:\n");
62     mostrar_matriz(B, A_filas, A_columnas);
63     printf("\nMatriz R:\n");
64     mostrar_matriz(R, A_filas, A_columnas);
65     break;
66
67 case 2:
68     printf("Multiplicacion de matrices");
69     printf("dimensiones A (CxR): ");
70     scanf("%dx%d", &A_filas, &A_columnas);
71
72     printf("dimensiones B (CxR): ");
73     scanf("%dx%d", &B_filas, &B_columnas);
74
75     printf("Matriz A:\n");
76     captura_matriz(A, A_filas, A_columnas);
77     printf("Matriz B:\n");
78     captura_matriz(B, B_filas, B_columnas);
79
80     multiplicar_matriz(A, B, R, A_filas, A_columnas, B_filas, B_columnas);
81
```

```
82     printf("\nMatriz A:\n");
83     mostrar_matriz(A, A_filas , A_columnas);
84     printf("\nMatriz B:\n");
85     mostrar_matriz(B, B_filas , B_columnas);
86     printf("\nMatriz R:\n");
87     mostrar_matriz(R, A_filas , B_columnas);
88     break;
89
90 case 3:
91     printf("Transpuesta de matrices\n\n");
92     printf("dimensiones (CxR): ");
93     scanf("%dx%d", &A_filas , &A_columnas);
94
95     printf("Matriz A:\n");
96     captura_matriz(A, A_filas , A_columnas);
97
98     transpuesta_matriz(A, R);
99     printf("\nMatriz A:\n");
100    mostrar_matriz(A, A_filas , A_columnas);
101    printf("Matriz R:\n");
102    mostrar_matriz(R, A_columnas, A_filas);
103    break;
104 }
105
106 return 0;
107 }
108
109 void multiplicar_matriz(int A[][8], int B[][8], int R[][8], int A_filas ,
110                        int A_columnas, int B_filas , int B_columnas) {
111     int n = 0;
112
113     if (A_columnas != B_filas) {
114         printf("numero incorrecto de filas y columnas");
115         exit(EXIT_SUCCESS);
116     }
117
118     n = A_columnas;
119
120     for (int r = 0; r < n; r++) {
121         for (int i = 0; i < A_filas; i++) {
122             for (int j = 0; j < B_columnas; j++) {
123                 R[i][j] += A[i][r] * B[r][j];
124             }
125         }
126     }
127 }
128
129 void transpuesta_matriz(int A[][8], int R[][8]) {
```

```
130     for (int i = 0; i < 8; i++) {
131         for (int j = 0; j < 8; j++) {
132             R[j][i] = A[i][j];
133         }
134     }
135 }
136
137 void sumar_matriz(int A[][8], int B[][8], int R[][8]) {
138     /* Matriz R es de resultado, se para como referencia para
139        que la funcion no use tanta memoria */
140     for (int i = 0; i < 8; i++) {
141         for (int j = 0; j < 8; j++) {
142             R[i][j] = A[i][j] + B[i][j];
143         }
144     }
145 }
146
147 void mostrar_matriz(int A[][8], size_t filas, size_t columnas) {
148     for (int i = 0; i < filas; i++) {
149         for (int j = 0; j < columnas; j++) {
150             printf("%d ", A[i][j]);
151         }
152         printf("\n");
153     }
154 }
155
156 void captura_matriz(int A[][8], size_t filas, size_t columnas) {
157     for (int i = 0; i < filas; i++) {
158         for (int j = 0; j < columnas; j++) {
159             mostrar_matriz(A, filas, columnas);
160             printf("[%d][%d]: ", i + 1, j + 1);
161             scanf("%d", &A[i][j]);
162             printf("\n");
163         }
164     }
165 }
166
167 void inicializar_matriz(int A[][8]) {
168     for (int i = 0; i < 8; i++) {
169         for (int j = 0; j < 8; j++) {
170             A[i][j] = 0;
171         }
172     }
173 }
174
175 void matriz_valida(size_t filas, size_t columnas) {
176     /* no hay matrices de dimensiones negativas o
177        menores a 2 */
```

```

178     if (filas < 2 || columnas < 2) {
179         printf("No es posible computar matrices con dimensión menor a 2\n");
180         exit(EXIT_SUCCESS);
181     }
182 }
183
184 int seleccionar_operacion() {
185     /* operaciones */
186     int op;
187
188     do {
189         printf("Operaciones:\n");
190         printf("- 1 :: Suma\n");
191         printf("- 2 :: Multiplicacion\n");
192         printf("- 3 :: Transpuesta\n");
193         printf("- 0 :: Salir\n");
194         printf("_____ \n");
195         printf("op: ");
196         scanf("%d", &op);
197         printf("\n");
198     } while (op < 0 || op > 3);
199
200     return op;
201 }

```

```

1  / Escriba un código que solicite 2 números y los reste. Desplegar un 1 si el
2  / resultado fue negativo o un 0 en caso contrario.
3
4  INPUT                                / Captura el valor de X
5  Store X
6  INPUT                                / Captura un valor de Y
7  Store Y
8
9  load X                               / Carga el valor de X en el acumulador
10 Subt Y                               / Resta el valor de Y a X
11
12                                     / Para ese punto, el valor en AC es 'X-Y'
13
14 Skipcond 000                         / si AC es mayor o igual a 0
15 clear                               / el valor en AC se vuelve 0
16
17 Skipcond 400                         / si es diferente a 0
18 Load verdadero                     / el valor en AC se vuelve 1
19
20 Output
21 Halt
22
23 X, DEC 0

```

```
24 | Y, DEC 0
25 | verdadero , dec 1

1 | / ESCRIBA UN CÓDIGO QUE SOLICITE 2 NÚMEROS Y LOS RESTE.
2 | / DESPLEGAR UN 1 SI EL RESULTADO FUE NEGATIVO O UN 0 EN CASO
3 | / CONTRARIO.
4 |
5 | INPUT                / CAPTURA EL VALOR DE X
6 | STORE X
7 | INPUT                / CAPTURA UN VALOR DE Y
8 | STORE Y
9 |
10 | LOAD X               / CARGA EL VALOR DE X EN EL ACUMULADOR
11 | SUBT Y               / RESTA EL VALOR DE Y A X
12 |
13 |                     / EL VALOR EN AC ES 'X-Y'
14 |
15 | SKIPCOND 000         / SI AC ES MAYOR O IGUAL A 0
16 | CLEAR               / EL VALOR EN AC SE VUELVE 0
17 |
18 | SKIPCOND 400         / SI ES DIFERENTE A 0
19 | LOAD VERDADERO      / EL VALOR EN AC SE VUELVE 1
20 |
21 | OUTPUT
22 | HALT
23 |
24 | X, DEC 0
25 | Y, DEC 0
26 | VERDADERO, DEC 1
```