Universidad de Guanajuato División de Ingenierías
Campus Irapuato Salamanca (DICIS)

Algoritmos y estructura de datos

Carlos Hugo García Capulín

Tarea No. 8

Reporte Memoria dinámica y estructura de datos

Jair Chávez Islas

18/Octubre/2021

Problema

Sacar el resultado del producto de un par de matrices con memoria dinámica

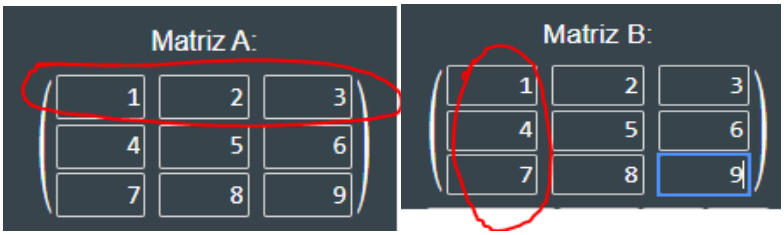
¿Qué es una matriz?

Una matriz es un conjunto de números ordenados en una estructura de filas y columnas, por ejemplo.

$$B=\begin{pmatrix} 1 & 5 & 2.3 \\ -2 & 0 & 0 \\ 3 & -1.1 & 2 \end{pmatrix}$$

Multiplicación de matrices

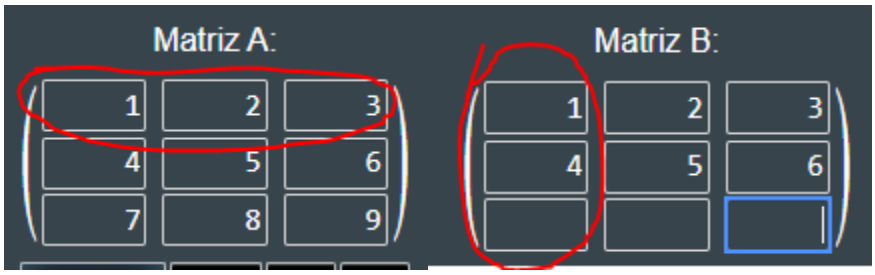
Para que se pueda dar la multiplicación de matrices también se necesita seguir una regla y es que, el número de filas de la matriz A, debe ser igual al número de columnas de la matriz B, con el siguiente ejemplo tenemos que,



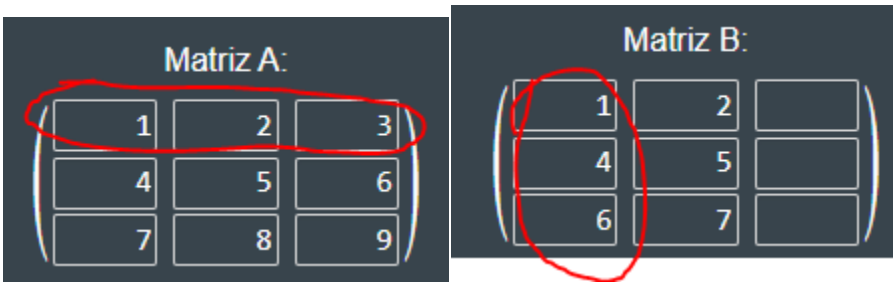
La columna de la matriz A {1,2,3} se multiplica con la fila de la matriz B{1,4,7} de la siguiente manera 1*1; 2*4; 3*7; los resultados se suman y en la matriz C, se pone el resultado en la casilla en la que coincidan la fila y la columna si es que estuviesen en la misma matriz, en este mismo ejemplo sería 1+8+21=30 y se pone en la primera casilla, y así con todos, hasta tener la matriz completa.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$$

De ser distinto, tenemos el siguiente ejemplo



Queda un número sin multiplicarse y es por eso que no se podría llevar a cabo esta multiplicación.

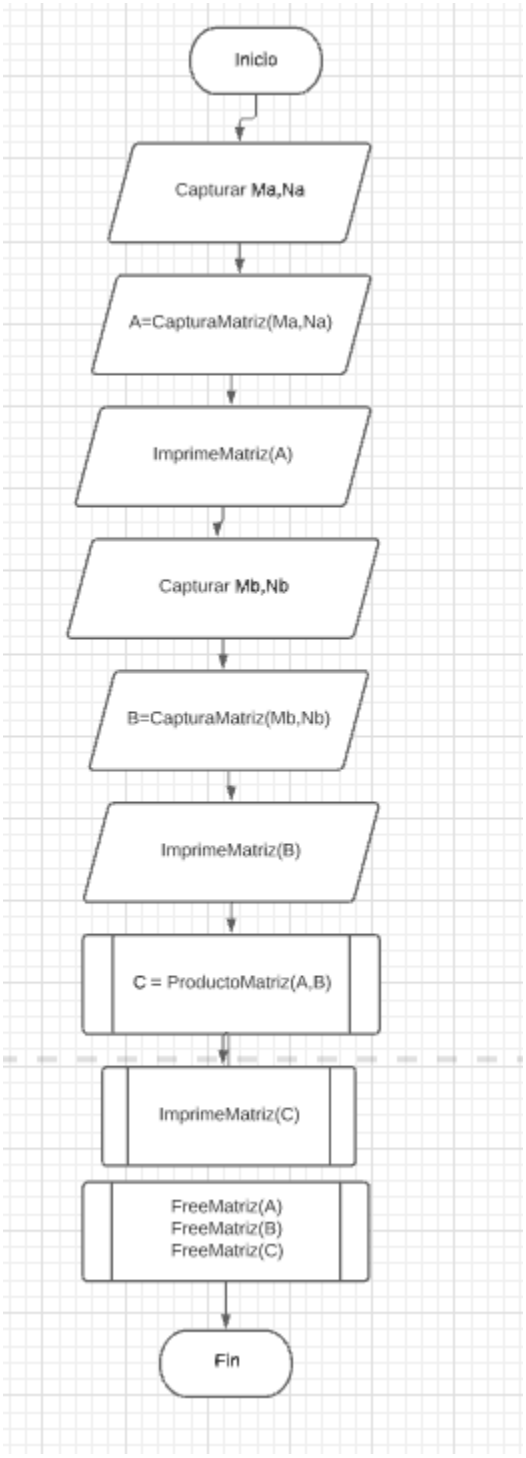


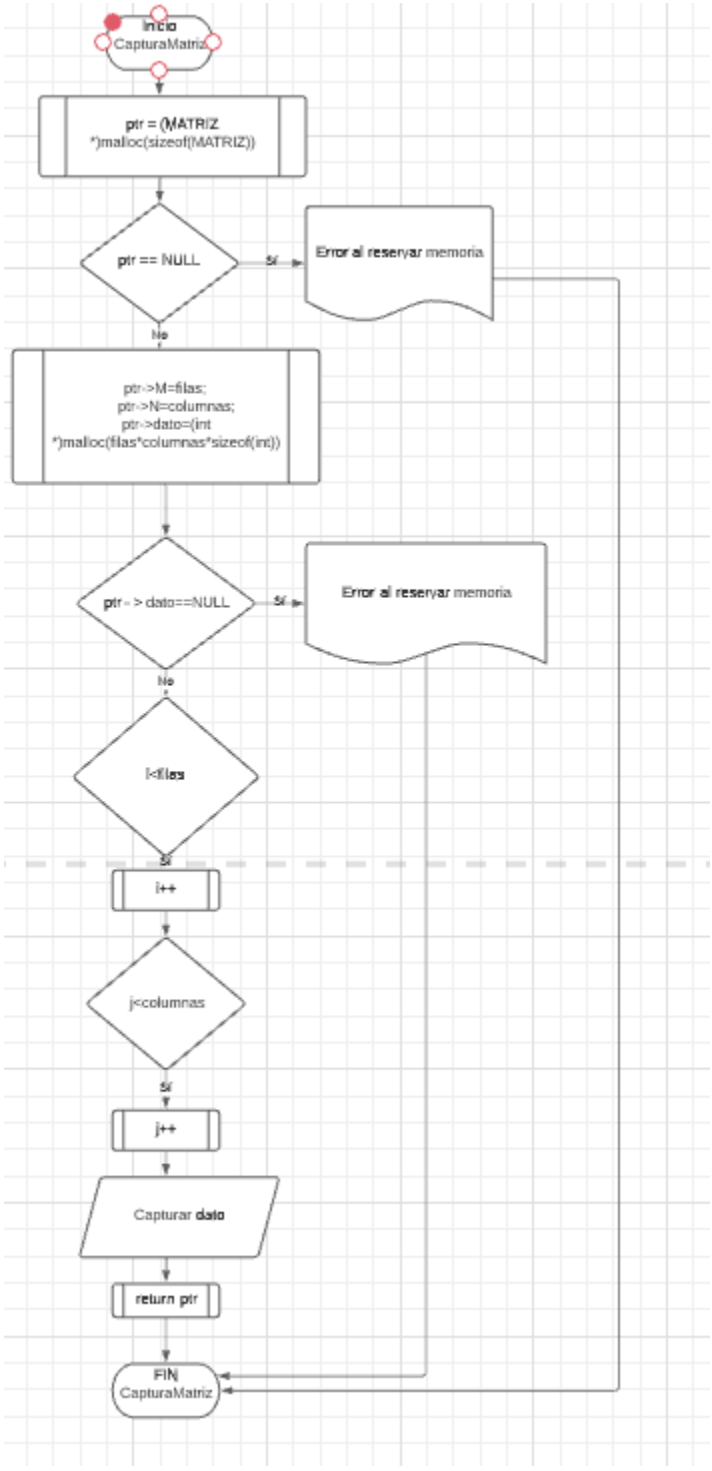
En este ejemplo sí que se puede multiplicar, dando como resultado la siguiente matriz.

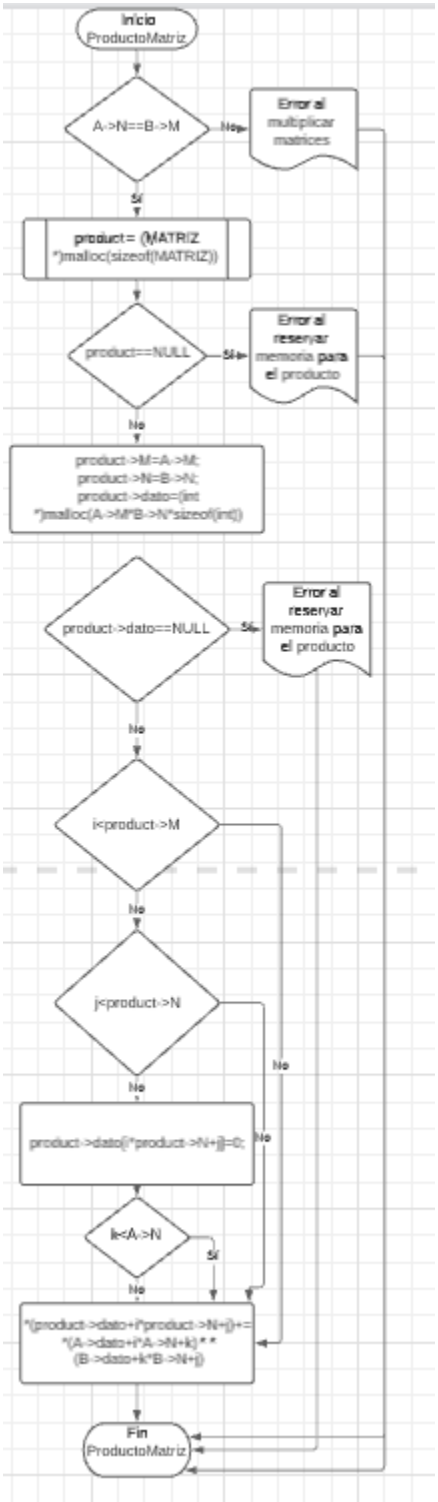
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 4 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix} 27 & 33 \\ 60 & 75 \\ 93 & 117 \end{pmatrix}$$

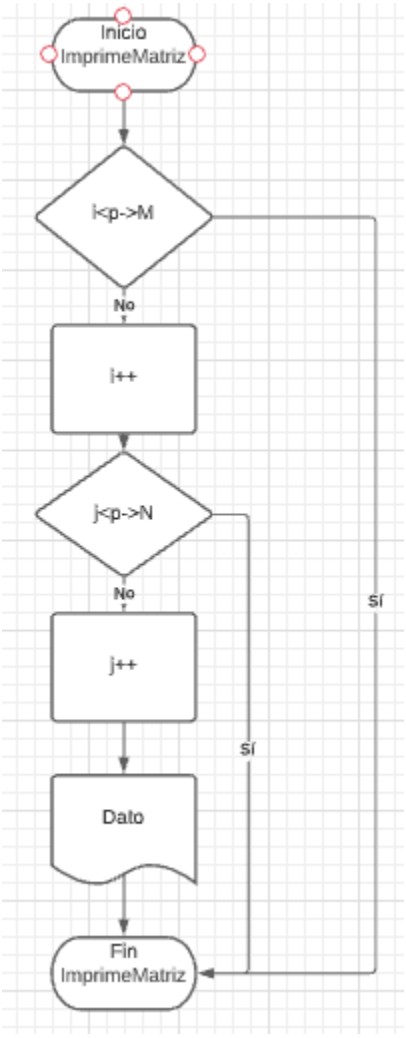
Solución implementada

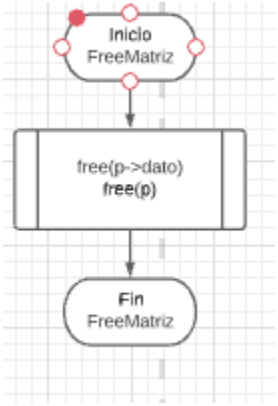
Diagrama de flujo




Inicio
CapturaMatriz
ptr = (MATRIZ
*)malloc(sizeof(MATRIZ))
ptr == NULL
Si
Error al reservar memoria
No
ptr->M=filas;
ptr->N=columnas;
ptr->dato=(int
*)malloc(filas*columnas*sizeof(int))
ptr->dato==NULL
Si
Error al reservar memoria
No
i<filas
Si
i++
j<columnas
Si
j++
Capturar dato
return ptr
FIN
CapturaMatriz


Inicio
ProductoMatriz
A->N==B->M
No
Error al multiplicar matrices
Si
product = (MATRIZ *)malloc(sizeof(MATRIZ))
product==NULL
Si
Error al reservar memoria para el producto
No
product->M=A->M;
product->N=B->N;
product->dato=(int *)malloc(A->M*B->N*sizeof(int))
product->dato==NULL
Si
Error al reservar memoria para el producto
No
i<product->M
No
j<product->N
No
No
product->dato[i*product->N+j]=0;
No
k<A->N
Si
No
*(product->dato+i*product->N+j)+= *(A->dato+i*A->N+k) * * (B->dato+k*B->N+j)
Fin
ProductoMatriz


Inicio
ImprimeMatriz
i<p->M
No
i++
j<p->N
No
j++
Dato
Fin
ImprimeMatriz
sí
sí


Inicio
FreeMatriz
free(p->dato)
free(p)
Fin
FreeMatriz

Código comentado

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct //Se crea una estructura llamada matriz
{
    //Esta tiene como miembros, M,N y dato
    unsigned int M;
    unsigned int N;
    int *dato;
}MATRIZ;

//Aqui se ponen los prototipos de las funciones usadas en el programa
MATRIZ* CapturaMatriz(unsigned int filas, unsigned int columnas);
MATRIZ* ProductoMatriz(MATRIZ *A, MATRIZ *B);
void ImprimeMatriz(MATRIZ *p);
void FreeMatriz(MATRIZ *p);

//Se inicializa la funcion principal
int main()
{
    //Se declaran las variables locales de la funcion
    unsigned int Ma, Na, Mb, Nb;
    int *A, *B, *C;

    //Se pregunta al usuario el numero de filas de A y se guarda en Ma
    printf("\nNumero de filas de A: ");
    scanf("%u",&Ma);
    //Se pregunta al usuario el numerod e columnas de A y se guarda en Na
    printf("\nNumero de columnas de A: ");
    scanf("%u",&Na);
    //Se guarda en la variable A la funcion CapturaMatriz aplicandola a las variables Ma y Na
    A=CapturaMatriz(Ma,Na);
    printf("\n");

    //Se una la funcion imprimematriz aplicandola a A para imprimir el resultado
    ImprimeMatriz(A);

    printf("\n");
    //Se pregunta al usuario el numero de filas de B y se guarda en Mb
    printf("\nNumero de filas de B: ");
    scanf("%u",&Mb);
    //Se pregunta al usuario el numerod e columnas de B y se guarda en Nb
    printf("\nNumero de columnas de B: ");
    scanf("%u",&Nb);
    printf("\n");
    //Se guarda en la variable B la funcion CapturaMatriz aplicandola a las variables Mb y Nb
    B=CapturaMatriz(Mb,Nb);
    printf("\n");
    //Se una la funcion imprimematriz aplicandola a B para imprimir el resultado
    ImprimeMatriz(B);

```

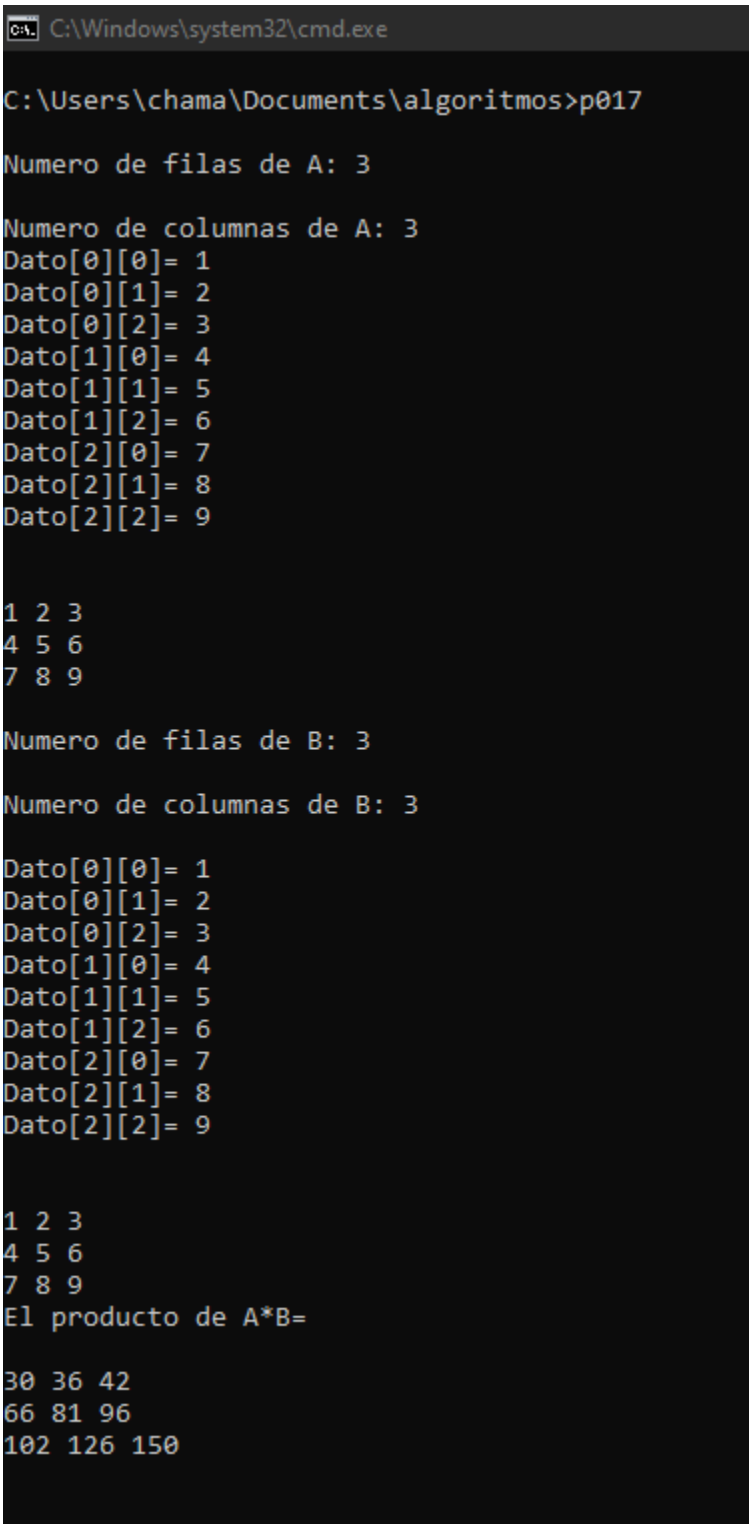
```
51
52      //Se guarda en la variable C el resultado de la funcion productomatriz aplciandola a A y B
53      C=ProductoMatriz(A,B);
54      printf("\nEl producto de A*B= \n");
55      //Se imprime el resultado del producto de las 2 matrices
56      ImprimeMatriz(C);
57      printf("\n");
58
59      //Se libera la memoria con una funcion para la matriz A, B y C
60      FreeMatriz(A);
61      FreeMatriz(B);
62      FreeMatriz(C);
63
64      printf("\n");//Aqui solo se imprime un salto d elinea
65
66      return 0;//se retorna el valor de 0 para verificar que haya finalizado sin problemas
67  }
68
69  //Se inicia la funcion Free Matriz, esta la usamos en la funcion main para liberar la memoria de las matrices
70  void FreeMatriz(MATRIZ *p)
71  {
72      free(p->dato);
73      free(p);
74  }
75
76  //Se inicia la funcion producto matriz
77  MATRIZ* ProductoMatriz(MATRIZ *A, MATRIZ *B)
78  {
79      //Se declaran las varibales locales de la funcion
80      unsigned int i,j,k;
81
82      //Se declara la variable product de tipo MATRIZ
83      MATRIZ *product=NULL;
84      //Para que se pueda sacar el producto, las columnas de A, deben ser igual a las filas de B
85      if(A->N==B->M)
86      {
87          //Se reserva memoria para la variable product
88          product = (MATRIZ *)malloc(sizeof(MATRIZ));
89          if(product==NULL)//Si no se puede reservar la memoria, te arroja el siguiente mensaje
90          {
91              printf("Error al reservar memoria para el producto \n");
92              exit(0);
93          }
94              product->M=A->M;
95              product->N=B->N;
96
97              //Reservar la memoria para almacenar los datos de la matriz MxN
98              product->dato=(int *)malloc(A->M*B->N*sizeof(int));
99          if(product->dato==NULL)
100         {
101             //Si no se puede reservar la memoria, te arroja el siguiente mensaje
```

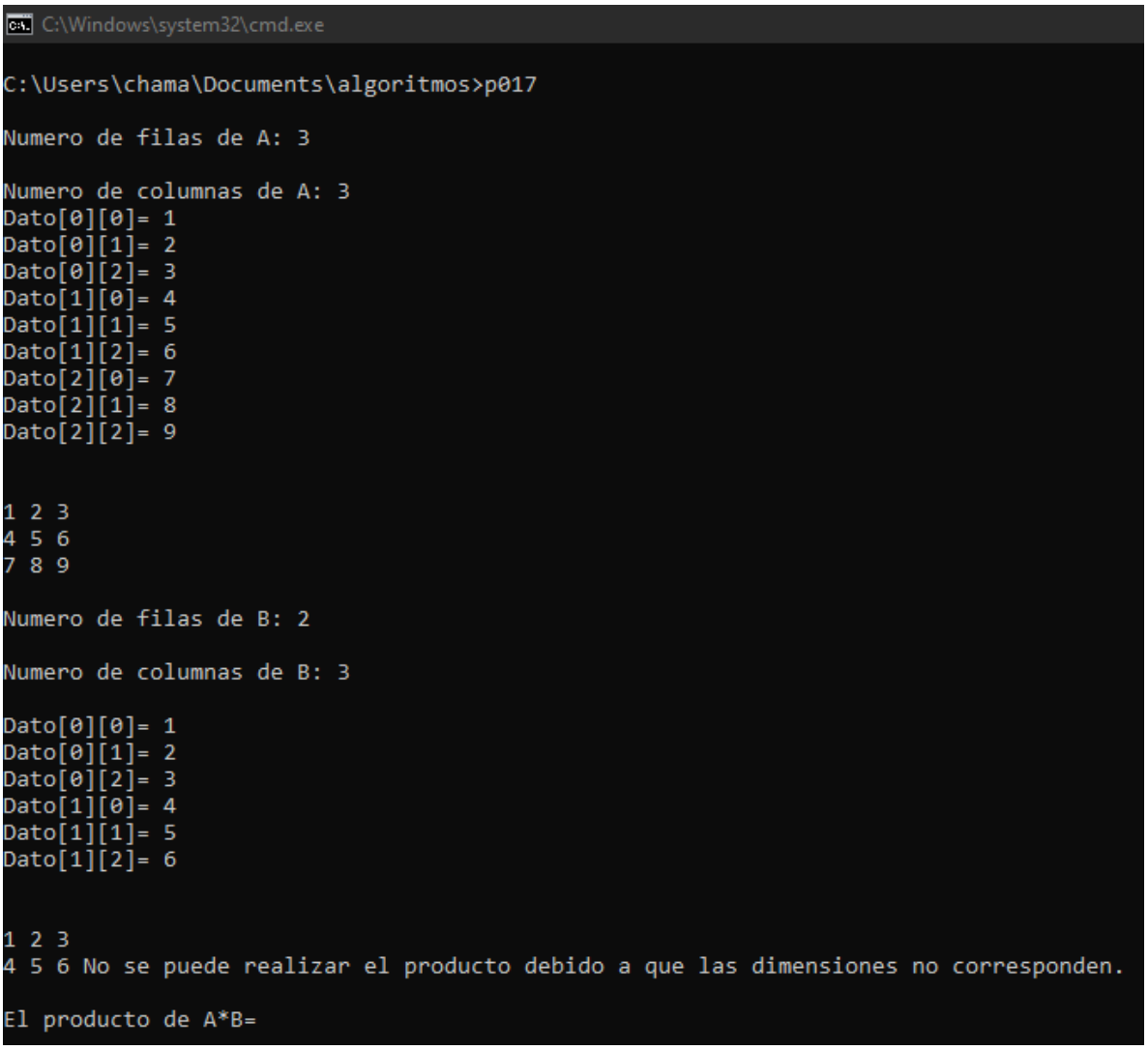
```
            //Si no se puede reservar la memoria, te arroja el siguiente mensaje
            printf("Error al reservar la memoria para los datos del producto MATRIZ. ");
            exit(0);
        }
        //En estos ciclos for, es donde nos empieza a hacer las operaciones correspondientes para sacar el producto
        for(i=0; i<product->M;i++)
            for(j=0;j<product->N;j++)
            {
                product->dato[i*product->N+j]=0;
                for(k=0;k<A->N;k++)
                    *(product->dato+i*product->N+j)+= *(A->dato+i*A->N+k) * * (B->dato+k*B->N+j);
            }
    }
    //No se puede sacar el producto por que las columnas de A, no son iguales a las filas de B
    else
    printf("No se puede realizar el producto debido a que las dimensiones no corresponden. \n");

    //Se retorna la variable product
    return product;
}

//Se inicia la funcion CapturaMatriz
MATRIZ* CapturaMatriz(unsigned int filas, unsigned int columnas)
{
    //Se declara la variable ptr de tipo MATRIZ
    MATRIZ *ptr;
    unsigned int i, j;

    //Reservar memoria para almacenar una estructura MATRIZ
    ptr = (MATRIZ *)malloc(sizeof(MATRIZ));
    if (ptr==NULL)
    {
    //Si no se puede reservar la memoria, te arroja el siguiente mensaje
        printf("Error al reservar la memoria para la matriz. \n");
        exit(0);
    }
    //Accediendo a los miembros de la estructura mediante un apuntador
    ptr->M=filas;
    ptr->N=columnas;

    //Reservar la memoria para almacenar los datos de la matriz de MxN
    ptr->dato=(int *)malloc(filas*columnas*sizeof(int));
    if(ptr->dato==NULL)
    {
        //Si no se puede reservar la memoria, te arroja el siguiente mensaje
        printf("Error al reservar la memoria para los datos de la MATRIZ.");
        exit(0);
    }
    //Iniciamos con la captura de los datos
    for (i = 0; i < filas; i++)
    {
```

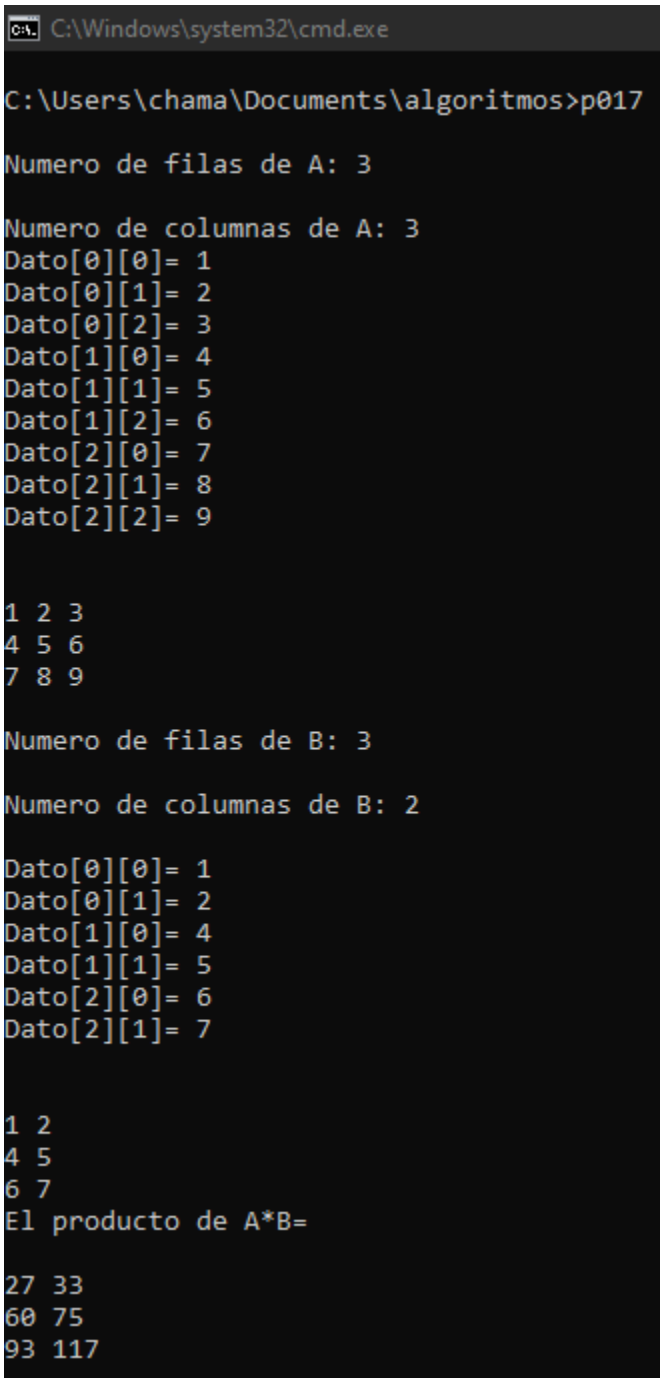
```
151     {
152         for(j=0; j<columnas; j++)
153         {
154             printf("Dato[%u][%u]= ",i,j);
155             scanf("%i",&ptr->dato[i*columnas+j]);
156         }
157     }
158     //Se retorna la variable ptr
159 
160     return ptr;
161 }
162 
163 //Se inicia la funcion imprimematriz
164 void ImprimeMatriz(MATRIZ *p)
165 {
166     //Se declaran las variables locales de la funcion
167     int i,j;
168     //Se abre un ciclo for
169     for(i=0;i<p->M;i++)
170     {
171         //SAlto de linea
172         printf("\n");
173         //Se abre otro ciclo for en el que ya se comenzaran a imprimir los datos de la matriz
174         for(j=0;j<p->N;j++)
175             printf("%i ", *(p->dato+i*p->N+j));
176     }
177 }
178 
```

Pruebas y resultados

Evidencia del programa




C:\Windows\system32\cmd.exe
C:\Users\chama\Documents\algoritmos>p017
Numero de filas de A: 3
Numero de columnas de A: 3
Dato[0][0]= 1
Dato[0][1]= 2
Dato[0][2]= 3
Dato[1][0]= 4
Dato[1][1]= 5
Dato[1][2]= 6
Dato[2][0]= 7
Dato[2][1]= 8
Dato[2][2]= 9
1 2 3
4 5 6
7 8 9
Numero de filas de B: 2
Numero de columnas de B: 3
Dato[0][0]= 1
Dato[0][1]= 2
Dato[0][2]= 3
Dato[1][0]= 4
Dato[1][1]= 5
Dato[1][2]= 6
1 2 3
4 5 6 No se puede realizar el producto debido a que las dimensiones no corresponden.
El producto de A*B=



Como vimos en el problema, no hay ninguna complicación si queremos multiplicar 2 matrices de las mismas dimensiones, porque el número de columnas de A es igual al de filas de B, pero en caso de no ser así, no se pueden multiplicar 2 matrices, y salieron los resultados previstos en cada uno de los ejemplos.

Conclusión: En este caso, ya habíamos visto un programa muy muy similar a este, solo que esta vez se usó memoria dinámica y esa es la ventaja, ya que, al haber visto el programa antes, ahora pude comprender los cambios de una manera más sencilla lo cual hizo que pudiese entender de mejor manera el tema.