**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

Alumno: Vargas Romero Erick Efraín

Profesor: Franco Martínez Edgardo Adrián

Unidad de aprendizaje: Estructuras de datos

Tarea 03: Programación con memoria dinámica

Secuencia: 1CM9

Semestre: 2016 – 2017 “A”

**Índice**

Problema 3

Pruebas 3

Prueba 1 3

Prueba 2 6

Prueba 3 6

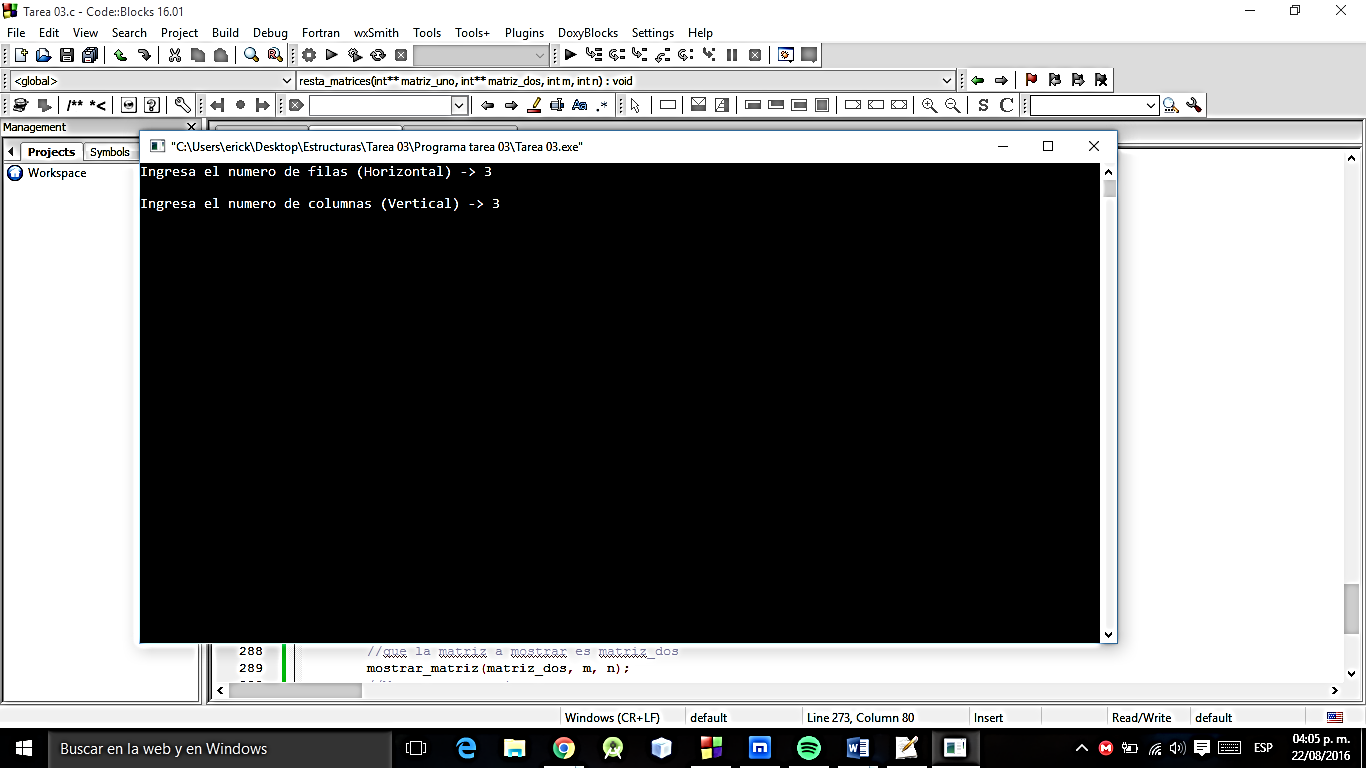
Implementación 9

# **Problema**

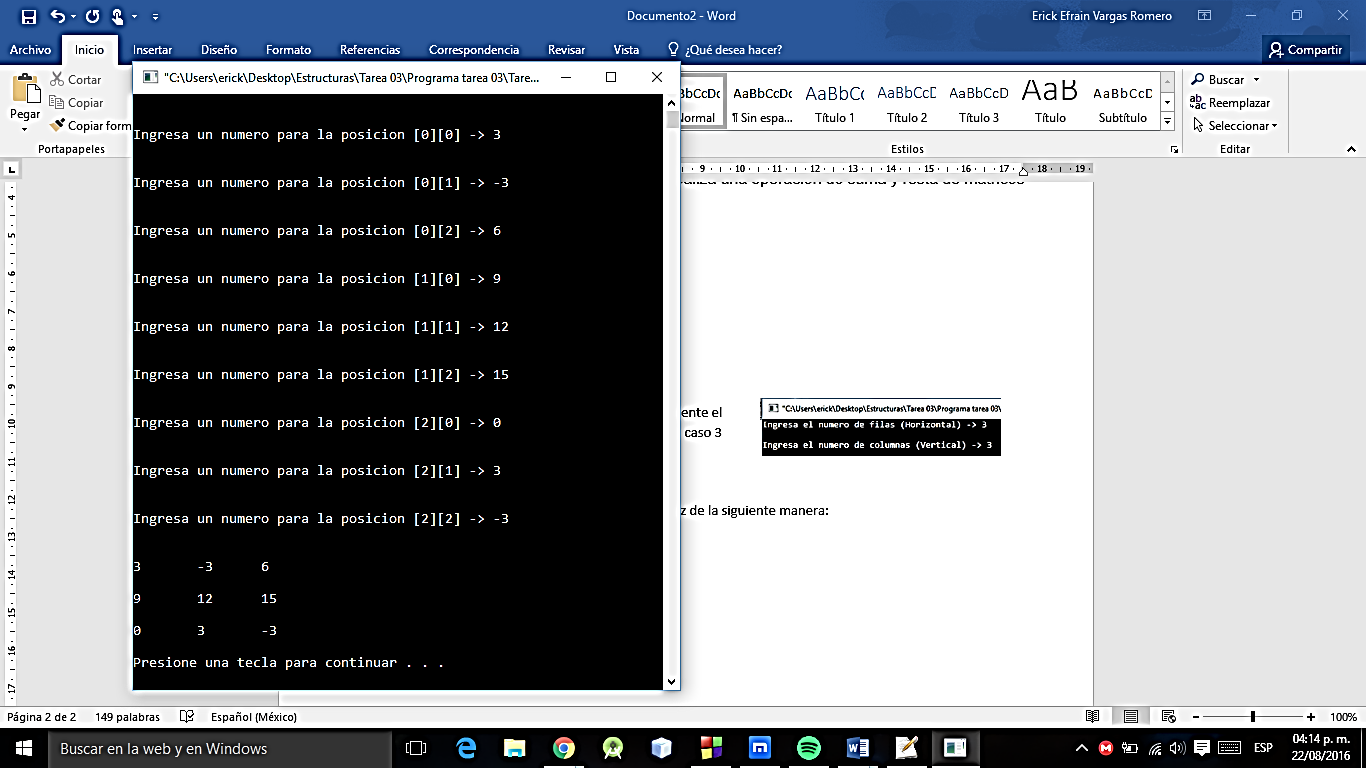
Programa de manera estructurada en ANSI C, un programa capaz de generar y recibir dos matrices dinámicas de tamaño m por n, donde m y n ingresan por la entrada estándar, además de los valores de las matrices, el programa realiza una operación de suma y resta de matrices que se muestra por la salida estándar.

# **Pruebas**

## **Prueba 1**

 En la primera prueba se ingresaron primeramente el número de filas y columnas que deseábamos, en este caso 3 filas y 3 columnas, como se muestra en la figura 1.1.

**Figura 1.1**

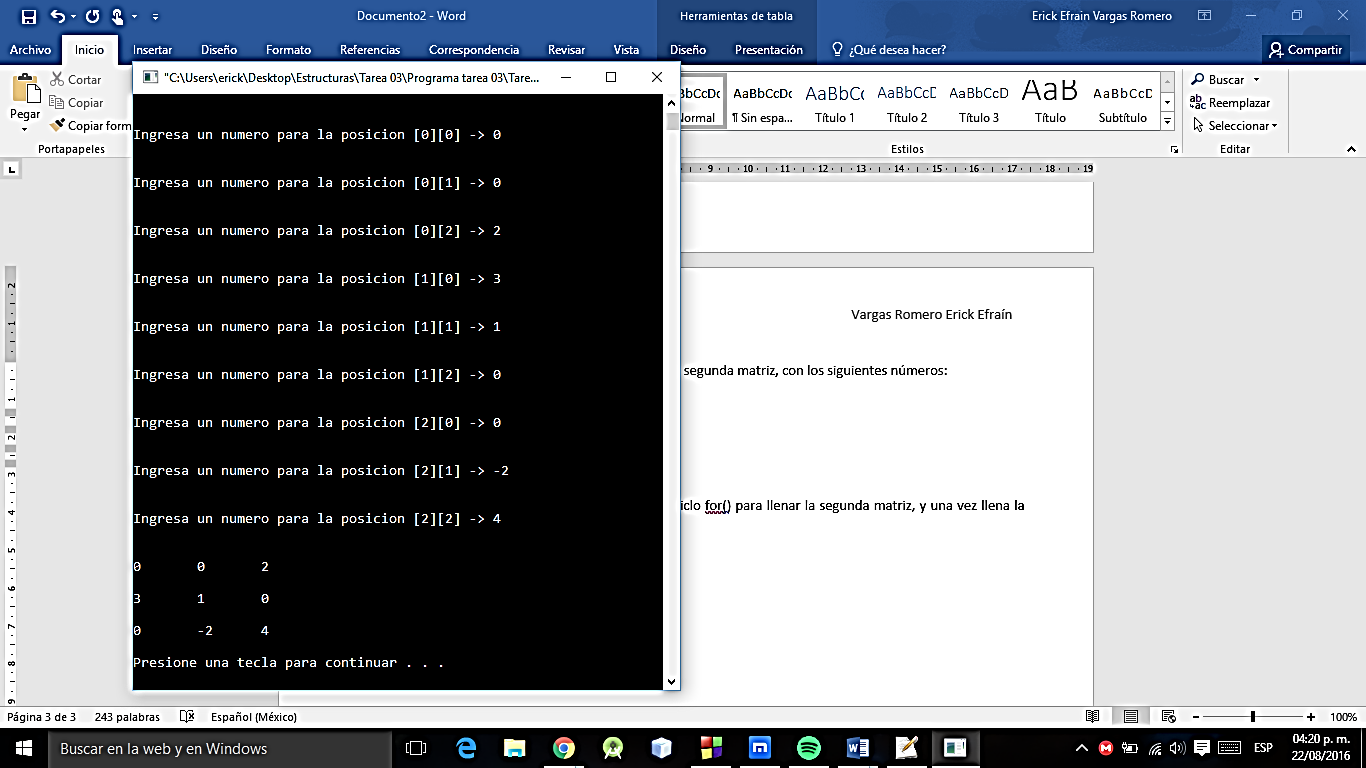


Posteriormente se hizo el llenado de la primera matriz de la siguiente manera:

|  |  |  |
| --- | --- | --- |
| Matriz uno | | |
| 3 | -3 | 6 |
| 9 | 12 | 15 |
| 0 | 3 | -3 |

En nuestro programa, utilizamos un ciclo de repetición, en específico el ciclo for(), para realizar el llenado de las matrices, una vez que se llenó la matriz, es mostrada al usuario, como en la figura 1.2

**Figura 1.2**

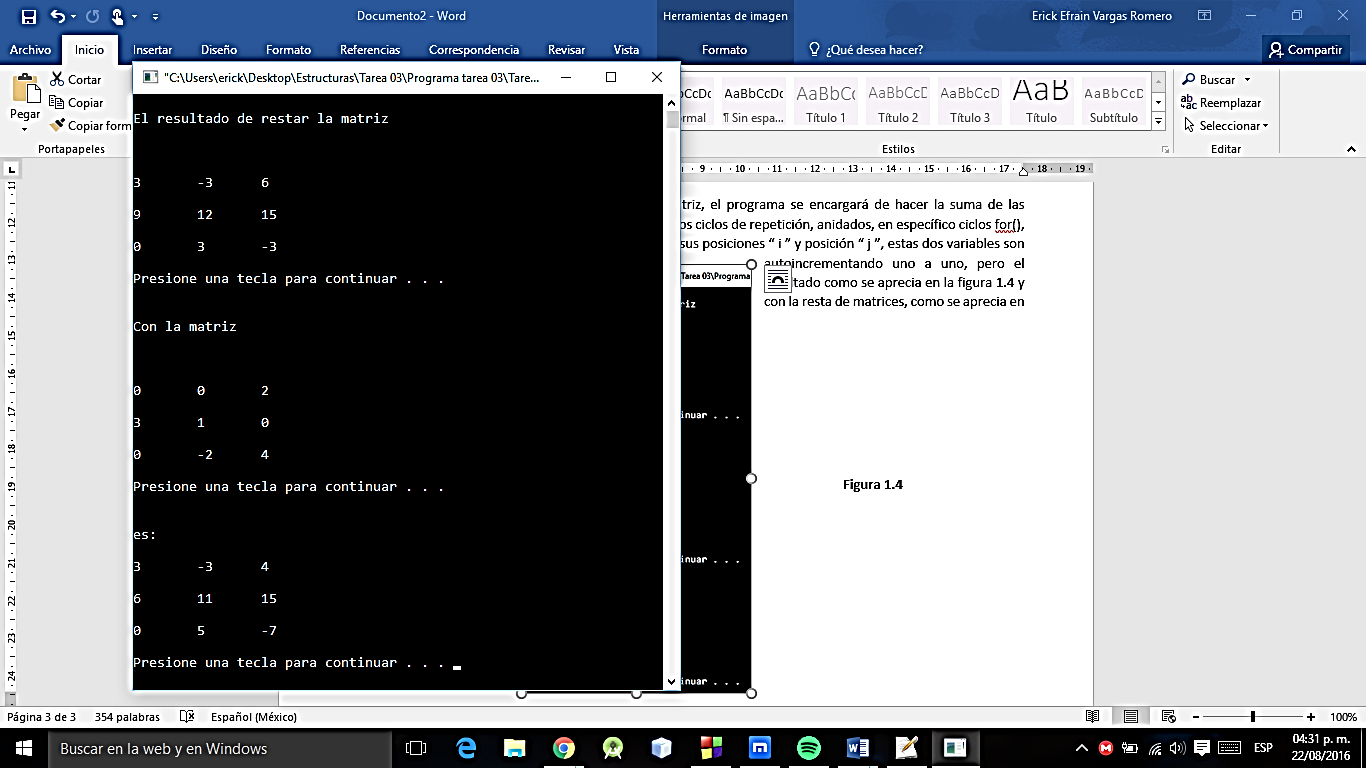
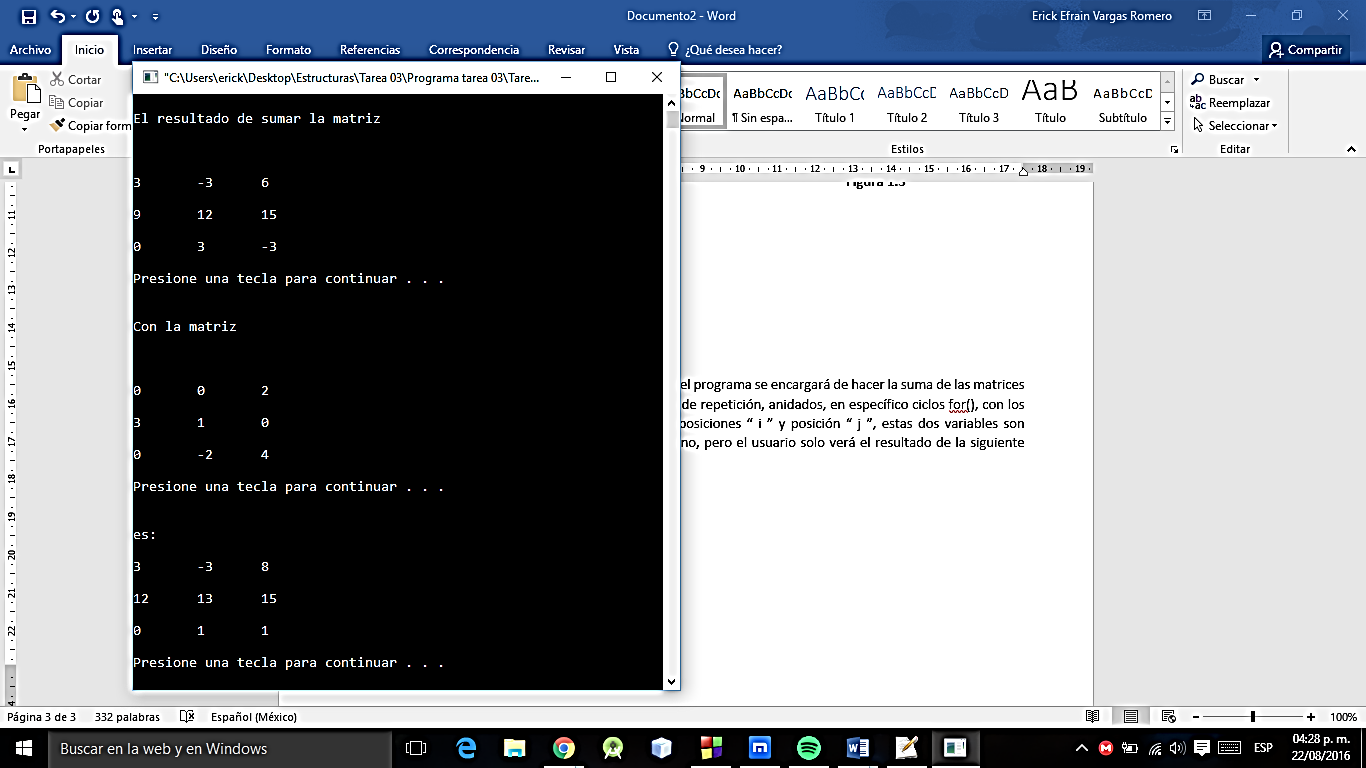


Una vez hecho lo anterior realizamos el llenado de la segunda matriz, con los siguientes números:

|  |  |  |
| --- | --- | --- |
| Matriz dos | | |
| 0 | 0 | 2 |
| 3 | 1 | 0 |
| 0 | -2 | 4 |

Al igual que el punto anterior, se utiliza un ciclo for() para llenar la segunda matriz, y una vez llena la segunda matriz, es mostrada al usuario, como se aprecia en la figura 1.3

**Figura 1.3**



Una vez, llenadas la primer y segunda matriz, el programa se encargará de hacer la suma de las matrices miembro a miembro, esto se hace usando dos ciclos de repetición, anidados, en específico ciclos for(), con los que se irá recorriendo la matriz uno y dos en sus posiciones “ i ” y posición “ j ”, estas dos variables son contadores, que se irán autoincrementando uno a uno, pero el usuario solo verá el resultado como se aprecia en la figura 1.4 y de igual manera ocurre con la resta de matrices, como se aprecia en la figura 1.5, además nótese que el resultado es correcto.

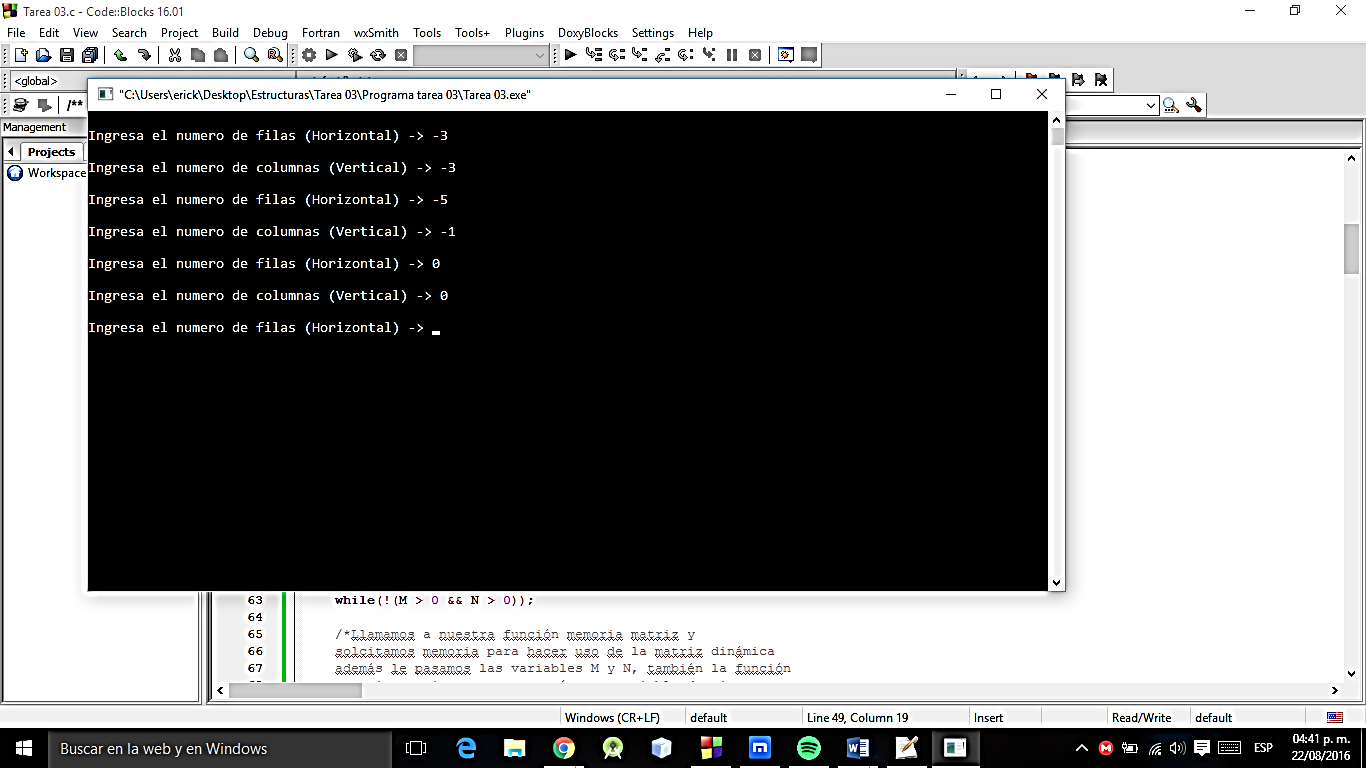
**Figura 1.5**

**Figura 1.4**

**Figura 1.4**

## **Prueba 2**

Ya hemos probado si el programa puede hacer sumas y restas de matrices 3 x 3 y debe de poder hacer sumas de matrices m x n, pero ¿qué pasará si ingresamos en el número de filas y columnas números negativos?

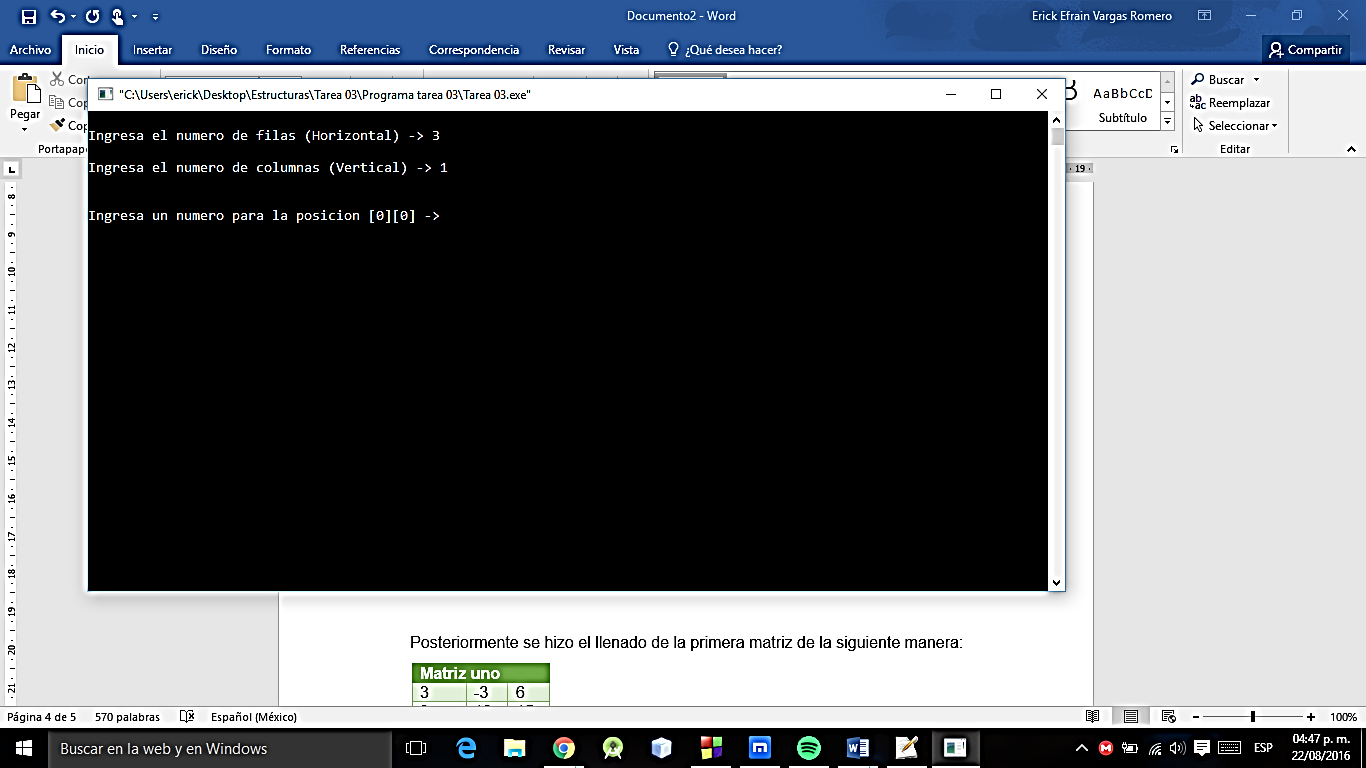


**Figura 2.1**

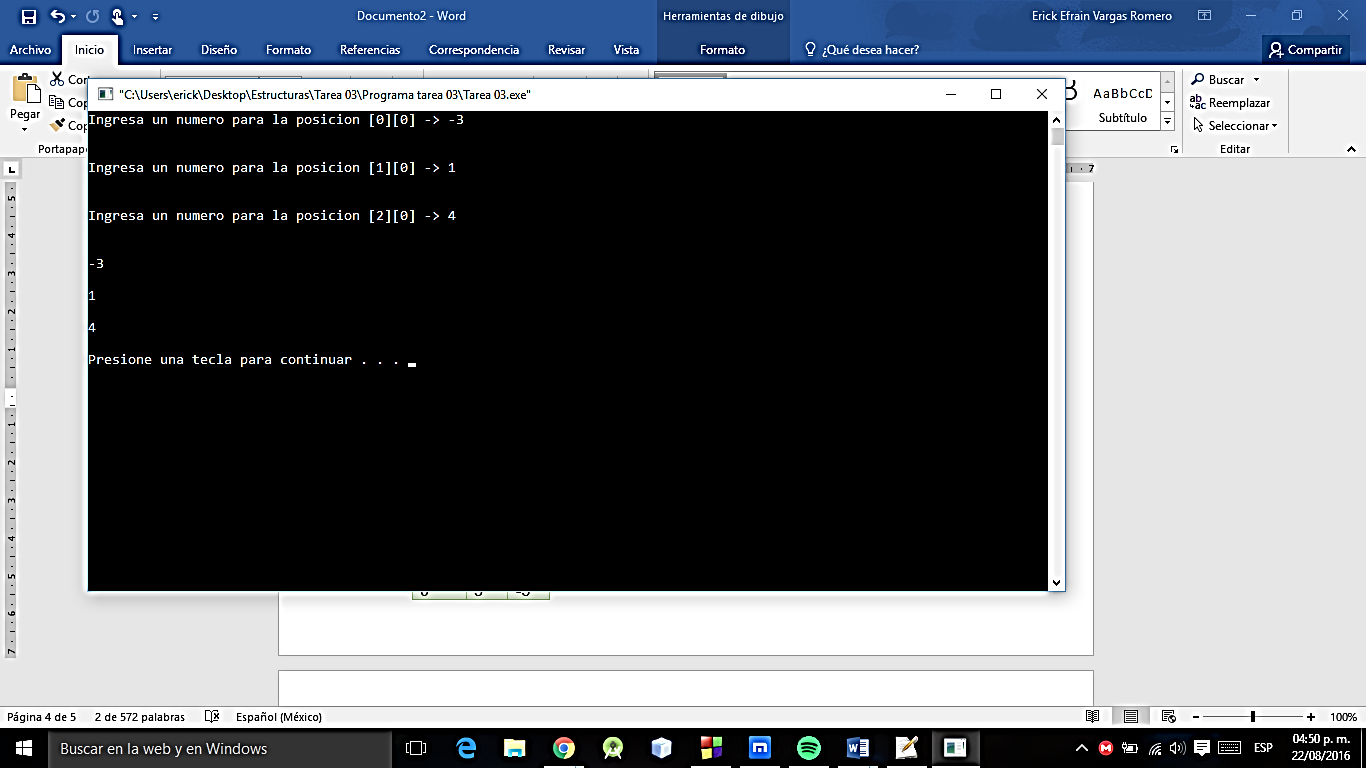
Como podemos apreciar en la figura 2.1, el programa no dejará continuar al usuario si en el número de filas y/o columnas se ingresa un número negativo o en su defecto sea cero, solo le dejará continuar si el número es mayor a cero.

## **Prueba 3**

Ya hemos probado lo que ocurriría si se tienen matrices, cuadradas, pero si no fuese el caso, ¿es posible realizar la suma y resta?

En la tercera prueba se ingresaron primeramente el número de filas y columnas que deseábamos, como ya hicimos la prueba con matrices cuadradas ingresaremos números diferentes en este caso 3 filas y 1 columna, como se muestra en la figura 3.1

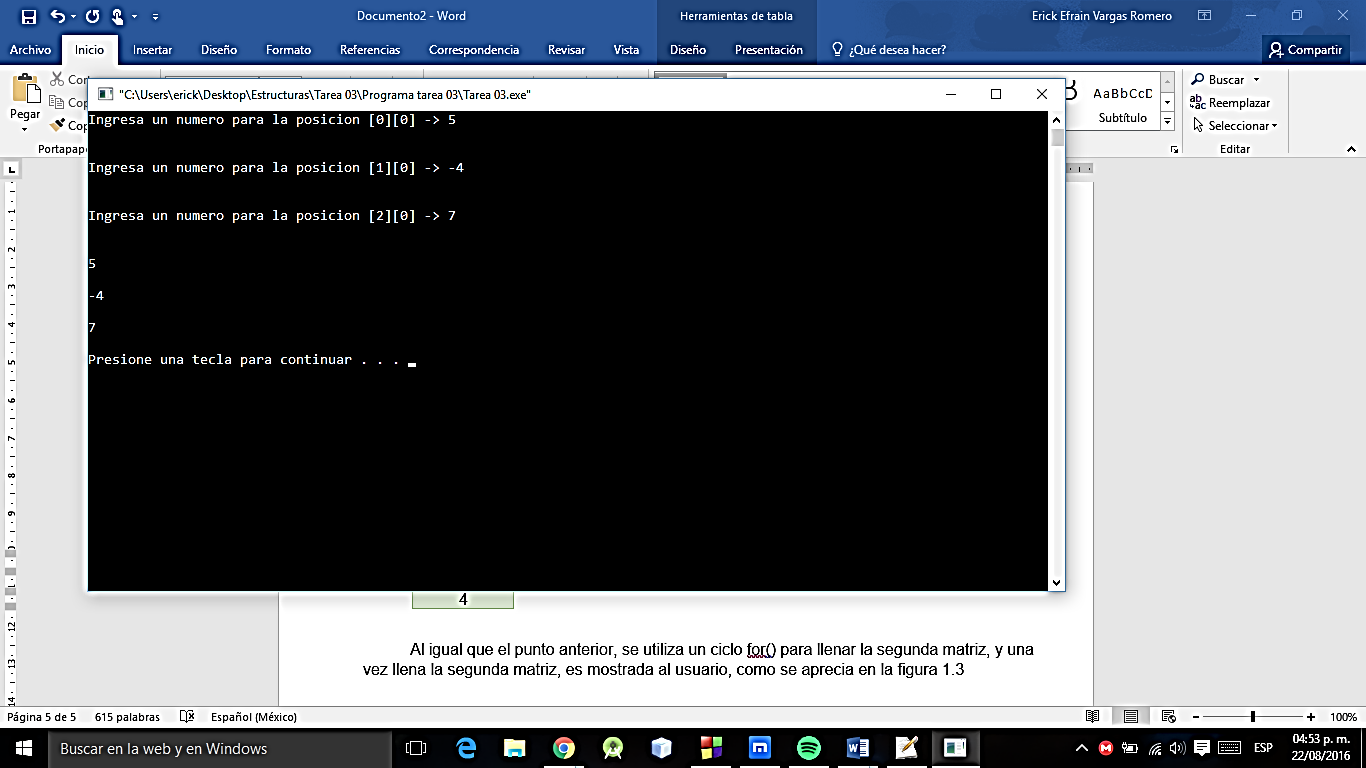
**Figura 3.1**

Posteriormente se hizo el llenado de la primera matriz de la siguiente manera:

|  |
| --- |
| Matriz uno |
| 3 |
| 1 |
| 4 |

Como se mencionó en la primera prueba, se ingresan los valores que almacenará la matriz uno y se mostrarán al usuario, como se muestra en la figura 3.2

**Figura 3.2**

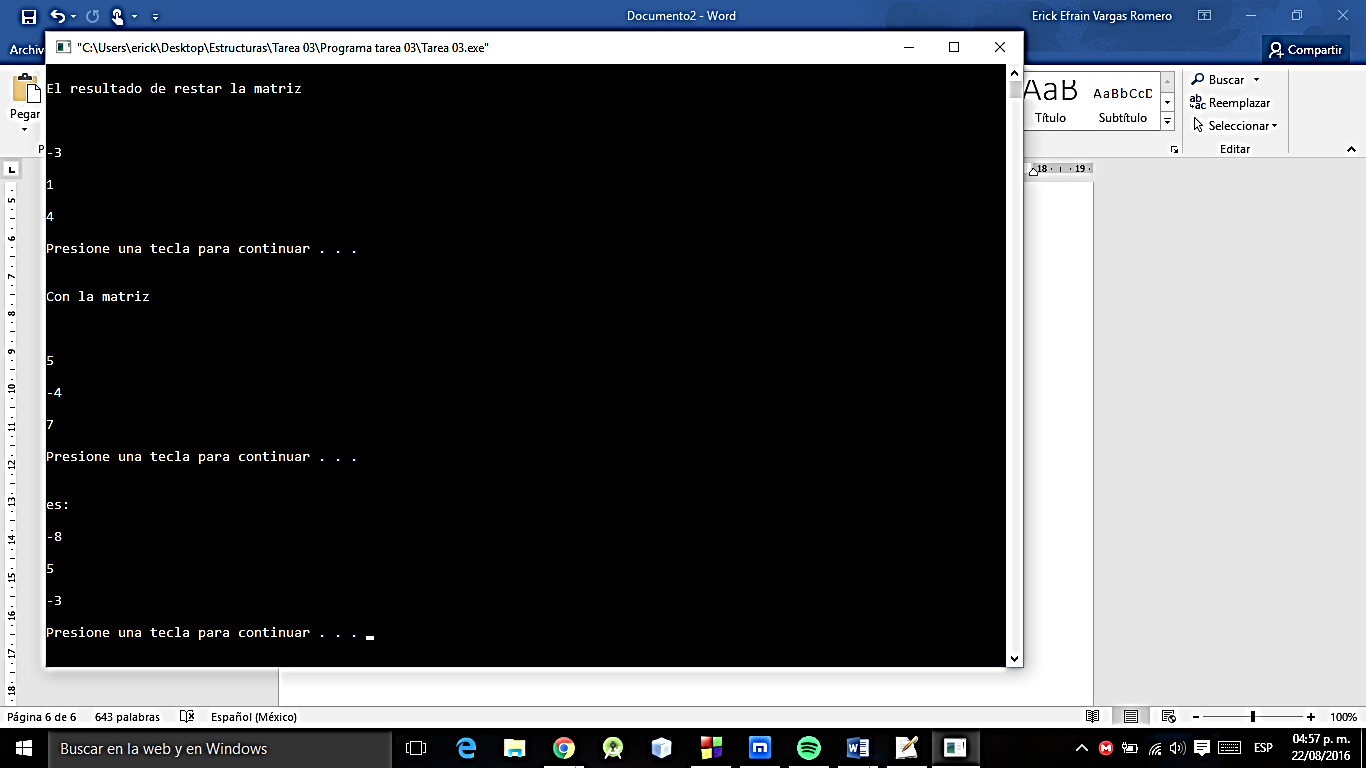
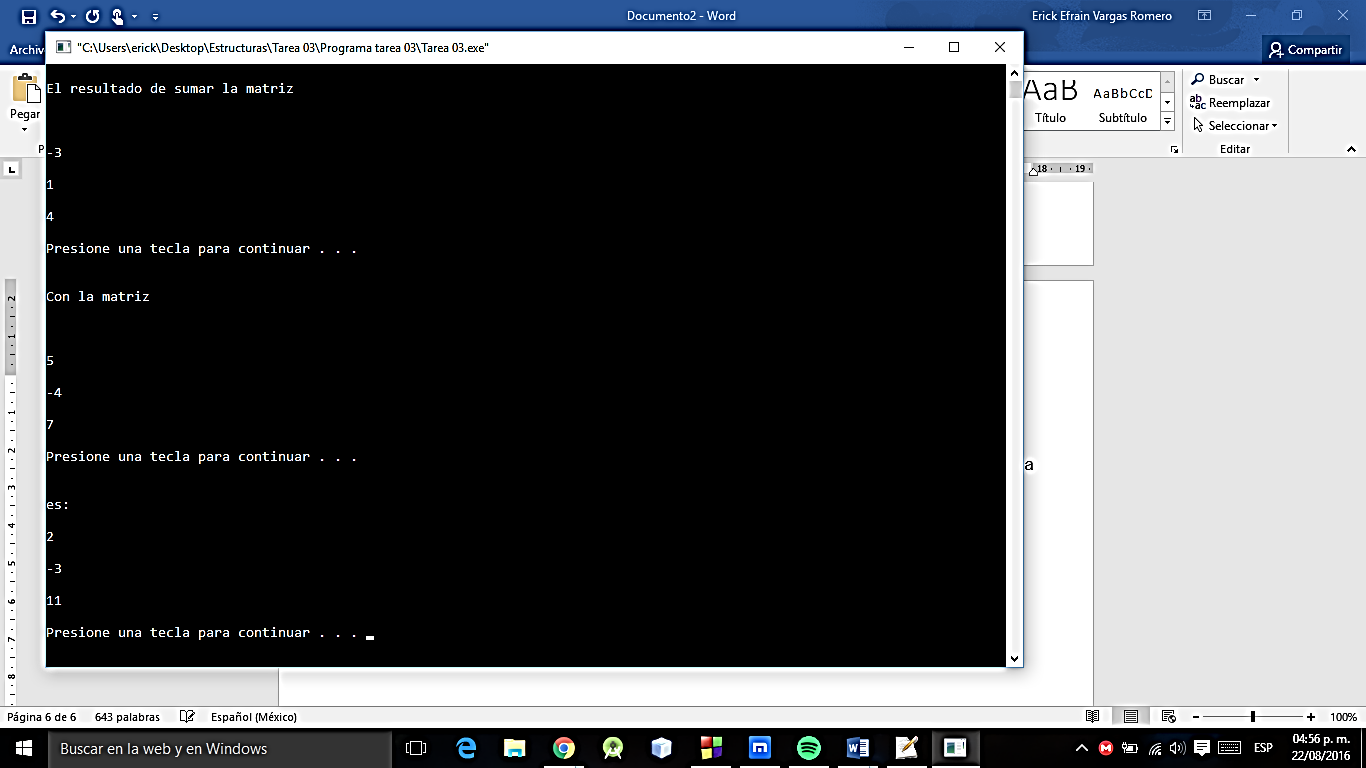
Una vez hecho lo anterior realizamos el llenado de la segunda matriz, con los siguientes números:

|  |
| --- |
| Matriz dos |
| 3 |
| 1 |
| 4 |

El llenado de la segunda matriz y de cualquier otra se hace de la misma manera, como se aprecia en la figura 3.3

**Figura 3.3**

Finalmente, el programa nos muestra el resultado de realizar la suma y la resta de ambas matrices como se muestra en las figuras 3.4, y 3.5, la primera con la suma y la segunda con la resta.



**Figura 3.5**

**Figura 3.4**

# **Implementación**

/\* Autor: Vargas Romero Erick Efraín, 22 de agosto 2016

Versión 1.0

Descripción: Programa que recibe el valor de filas y columnas para

crear dos matrices dinámicas, los valores de las filas y columnas

no serán menores a 1, el usuario ingresará los valores que

contendrá cada matriz dinámica en su posición i, j, finalmente

el programa, realizará la suma y resta de matrices dinámicas, esto

lo hace miembro a miembro, y almacenando estos valores en una matriz

resultante, que también será dinámica

Declaración de librerías, se incluyen las funciones estandar

de entrada y salida\*/

#include <stdio.h>

/\*Declaración de funciones

Aparta la memoria necesaria para nuestras matrices dinámicas, recibe

dos datos de tipo entero\*/

int **\*\***memoria\_matriz**(**int m**,** int n**);**

/\*Llenado de las matrices, esta función recibe tres datos uno de tipo

entero doble apuntador, y dos datos de tipo entero \*/

void llenar\_matriz**(**int **\*\***Matriz**,** int m**,** int n**);**

/\*Muestreo de lo que contiene la matriz, recibe 3 parámetros uno

de tipo entero doble apuntador y dos datos de tipo entero\*/

void mostrar\_matriz**(**int **\*\***Matriz**,** int m**,** int n**);**

/\*Función para realizar la suma de las matrices, recibe 4 parámetros

dos de tipo entero doble apuntador, y dos de tipo entero\*/

void suma\_matrices**(**int **\*\***matriz\_uno**,** int **\*\***matriz\_dos**,** int m**,** int n**);**

/\*Función para realizar la resta de las matrices, recibe 4 parámetros

dos de tipo entero doble apuntador, y dos de tipo entero\*/

void resta\_matrices**(**int **\*\***matriz\_uno**,** int **\*\***matriz\_dos**,** int m**,** int n**);**

//Programa principal

int main**(**void**)**

**{**

/\*Declaración de variables, dos de tipo entero doble apuntador

y dos de tipo entero, las dos primeras variables serán utilizadas

para la creación de nuestras matices dinámicas, y los últimos dos datos

son utilizados para especificar el número de fílas y columnas que

tendrán nuestras matrices dinámicas\*/

int **\*\***matriz\_uno**,** **\*\***matriz\_dos**,** M**,** N**;**

//Inicio de un ciclo de repetición

**do**

**{**

//Enviamos mensaje a pantalla pidiendo el número de filas

printf**(**"\nIngresa el numero de filas (Horizontal) -> "**);**

//Hacemos la lectura de las filas que desea el usuario

//Lo ingresado es almacenado en la variable M

scanf**(**"%d"**,** **&**M**);**

//Enviamos mensaje a pantalla pidiendo el número de columnas

printf**(**"\nIngresa el numero de columnas (Vertical) -> "**);**

//Hacemos la lectura de las columnas que desea el usuario

//Lo ingresado es almacenado en la variable N

scanf**(**"%d"**,** **&**N**);**

**}**

//Este ciclo se repetirá mientras las variables M y N

//No sean mayores a cero

**while(!(**M **>** 0 **&&** N **>** 0**));**

/\*Llamamos a nuestra función memoria matriz y

solcitamos memoria para hacer uso de la matriz dinámica

además le pasamos las variables M y N, también la función

memoria\_matriz, nos returnará otra variable de tipo

int\*\* que será guardado en la vriable matriz\_uno\*/

matriz\_uno **=** memoria\_matriz**(**M**,** N**);**

/\*Verificamos si el valor de nuestra variable\_matriz uno

no sea 0 o bien NULL (Este valor es la memoria que se le ha

asignado)\*/

**if(**matriz\_uno **==** **NULL)**

**{**

/\*Si el valor de la variable matriz\_uno es NULL, enviamos un mensaje

a pantalla y finalizamos el programa\*/

printf**(**"\n\nNo hay memoria disponible para la primer matriz...\n"**);**

**}**

/\*Si se le ha asignado memoria a matriz\_uno entonces...\*/

**else**

**{**

/\*Llamamos a la función llenar\_llenar matriz, como el nombre

lo indica, realizaremos el llenado de nuestra matriz dinámica

le pasamos variables, nuestra matriz\_uno de tipo int\*\* y

las filas y columnas de nuestra matriz\*/

llenar\_matriz**(**matriz\_uno**,** M**,** N**);**

/\*Invocamos a nuestra función mostrar\_matriz, como el nombre

lo indica, mostrará lo que previamente ingresamos a la matriz

en este caso, le pasamos 3 datos, el primero matriz\_uno, tipo

int\*\* y el número de filas y columnas\*/

mostrar\_matriz**(**matriz\_uno**,** M**,** N**);**

/\*Llamamos a nuestra función memoria matriz y

solcitamos memoria para hacer uso de la matriz dinámica

además le pasamos las variables M y N, también la función

memoria\_matriz, nos returnará otra variable de tipo

int\*\* que será guardado en la vriable matriz\_dos\*/

matriz\_dos **=** memoria\_matriz**(**M**,** N**);**

/\*Verificamos si el valor de nuestra variable\_matriz uno

no sea 0 o bien NULL (Este valor es la memoria que se le ha

asignado)\*/

**if(**matriz\_dos **==** **NULL)**

**{**

/\*Si el valor de la variable matriz\_uno es NULL, enviamos un mensaje

a pantalla y finalizamos el programa\*/

printf**(**"\n\nNo hay memoria disponible para la segunda matriz...\n"**);**

**}**

/\*Si se le ha asignado memoria a matriz\_uno entonces...\*/

**else**

**{**

/\*Llamamos a la función llenar\_llenar matriz, como el nombre

lo indica, realizaremos el llenado de nuestra matriz dinámica

le pasamos variables, nuestra matriz\_dos de tipo int\*\* y

las filas y columnas de nuestra matriz\*/

llenar\_matriz**(**matriz\_dos**,** M**,** N**);**

/\*Invocamos a nuestra función mostrar\_matriz, como el nombre

lo indica, mostrará lo que previamente ingresamos a la matriz

en este caso, le pasamos 3 datos, el primero matriz\_dos, tipo

int\*\* y el número de filas y columnas\*/

mostrar\_matriz**(**matriz\_dos**,** M**,** N**);**

/\*Esta función realizará la suma de matrices dinámicas

le pasamos 4 variables, matriz\_uno, matriz\_dos que son

de tipo int\*\*, también le pasamos M y N que son el número

de filas y columnas\*/

suma\_matrices**(**matriz\_uno**,** matriz\_dos**,** M**,** N**);**

/\*Esta función realizará la resta de matrices dinámicas

le pasamos 4 variables, matriz\_uno, matriz\_dos que son

de tipo int\*\*, también le pasamos M y N que son el número

de filas y columnas\*/

resta\_matrices**(**matriz\_uno**,** matriz\_dos**,** M**,** N**);**

**}**

**}**

**return** 0**;**

**}**

/\*Aparta la memoria necesaria para nuestras matrices dinámicas, recibe

dos datos de tipo entero, también tomemos en cuenta que esta función

nos returnará una variable de tipo int\*\* \*/

int **\*\***memoria\_matriz**(**int m**,** int n**)**

**{**

/\*Como la función nos returnará una variable de tipo int\*\*,

declaramos una variable del mismo tipo, también declaramos

otra variable "i", que usaremos como contador\*/

int **\*\***Matriz**,** i**;**

/\*Apartamos la memoria de nuestra matriz, en este caso, la memoria

para las filas, y es multiplicado el número de filas, o sea la

variable m por el valor que tenga un dato tipo int\*/

Matriz **=** malloc**(**m**\*sizeof(**int**));**

/\*Iniciamos un ciclo de repetición que terminará cuando nuestr variable

"i" sea menor que nuestra variable m, la variable i se incrementa 1 a 1\*/

**for(**i **=** 0**;** i**<**m**;** i**++)**

**{**

//Apartamos memoria para las columnas

Matriz**[**i**]** **=** malloc**(**n**\*sizeof(**int**));**

**}**

//Returnamos una variable de tipo int\*\*

**return** Matriz**;**

**}**

/\*Llenado de las matrices, esta función recibe tres datos uno de tipo

entero doble apuntador, y dos datos de tipo entero \*/

void llenar\_matriz**(**int **\*\***Matriz**,** int m**,** int n**)**

**{**

//Declaración de dos variables que serán contadores

int i**,** j**;**

//Ciclo de repetición, que hace el recorrido de las filas

**for(**i **=** 0**;** i**<**m**;** i**++)**

**{**

//Ciclo de repetición que hace el recorrido de las columnas

**for(**j **=** 0**;** j**<**n**;** j**++)**

**{**

//Enviamos mensaje solicitando que se asigne un valor en la

//posición mostrada

printf**(**"\n\nIngresa un numero para la posicion [%d][%d] -> "**,** i**,** j**);**

//Lectura del número ingresado y almacenado en la posición i, j

scanf**(**"%d"**,** **&**Matriz**[**i**][**j**]);**

**}**

**}**

**}**

/\*Muestreo de lo que contiene la matriz, recibe 3 parámetros uno

de tipo entero doble apuntador y dos datos de tipo entero\*/

void mostrar\_matriz**(**int **\*\***Matriz**,** int m**,** int n**)**

**{**

//Declaración de dos variables que serán contadores

int i**,** j**;**

//Ciclo de repetición, que hace el recorrido de las filas

**for(**i **=** 0**;** i**<**m**;** i**++)**

**{**

//Damos dos saltos de linea

printf**(**"\n\n"**);**

//Ciclo de repetición, que hace el recorrido de las columnas

**for(**j **=** 0**;** j**<**n**;** j**++)**

**{**

//Mostramos el valor que tiene la variable Matriz, en

//su posición i, j, además damos un tabulador entre cada

//dato mostrado

printf**(**"%d \t"**,** Matriz**[**i**][**j**]);**

**}**

**}**

//Damos dos saltos de linea

printf**(**"\n\n"**);**

//Pausamos el programa

system**(**"pause"**);**

**}**

/\*Función para realizar la suma de las matrices, recibe 4 parámetros

dos de tipo entero doble apuntador, y dos de tipo entero\*/

void suma\_matrices**(**int **\*\***matriz\_uno**,** int **\*\***matriz\_dos**,** int m**,** int n**)**

**{**

//Declaración de variables, dos contadores y una variable

//tipo int\*\* llamad matriz, suma

int i**,** j**,** **\*\***matriz\_suma**;**

/\*Como realizaremos una suma de matrices, debemos de crear

una tercer matriz, o matriz resultante\*/

matriz\_suma **=** memoria\_matriz**(**m**,** n**);**

//Verificamos si hay memoria disponible para la matriz resultante

**if(**matriz\_suma **==** **NULL)**

**{**

//Si no hay memoria, enviamos un mensaje

printf**(**"\n\nNo hay memoria disponible para la matriz resultante...\n"**);**

**}**

**else**

**{**

//Si la memoria es suficiente

//limpiamos pantalla

system**(**"cls"**);**

//Enviamos mensaje

printf**(**"\nEl resultado de sumar la matriz \n\n"**);**

//Invocamos a la función mostrar matriz, e indicamos

//que la matriz a mostrar es matriz\_uno

mostrar\_matriz**(**matriz\_uno**,** m**,** n**);**

//Enviamos mensaje

printf**(**"\n\nCon la matriz \n\n"**);**

//Invocamos a la función mostrar matriz, e indicamos

//que la matriz a mostrar es matriz\_dos

mostrar\_matriz**(**matriz\_dos**,** m**,** n**);**

//Mostramos mensaje

printf**(**"\n\nes: "**);**

//iniciamos un ciclo de repetición para las filas

**for(**i **=** 0**;** i**<**m**;** i**++)**

**{**

//Iniciamos un segundo ciclo de repetición para columnas

**for(**j **=** 0**;** j**<**n**;** j**++)**

**{**

//Realizamos la suma entre matrices y almacenamos en

//nuestra matriz resultante

matriz\_suma**[**i**][**j**]** **=** matriz\_uno**[**i**][**j**]** **+** matriz\_dos**[**i**][**j**];**

**}**

**}**

//mostramos nuestra matriz resultante usando la

//función mostrar\_matriz

mostrar\_matriz**(**matriz\_suma**,** m**,** n**);**

**}**

**}**

/\*Función para realizar la resta de las matrices, recibe 4 parámetros

dos de tipo entero doble apuntador, y dos de tipo entero\*/

void resta\_matrices**(**int **\*\***matriz\_uno**,** int **\*\***matriz\_dos**,** int m**,** int n**)**

**{**

//Declaración de variables, dos contadores y una variable

//tipo int\*\* llamad matriz, suma

int i**,** j**,** **\*\***matriz\_resta**;**

/\*Como realizaremos una suma de matrices, debemos de crear

una tercer matriz, o matriz resultante\*/

matriz\_resta **=** memoria\_matriz**(**m**,** n**);**

//Verificamos si hay memoria disponible para la matriz resultante

**if(**matriz\_resta **==** **NULL)**

**{**

//Si no hay memoria, enviamos un mensaje

printf**(**"\n\nNo hay memoria disponible para la matriz resultante...\n"**);**

**}**

**else**

**{**

//Si la memoria es suficiente

//limpiamos pantalla

system**(**"cls"**);**

//Enviamos mensaje

printf**(**"\nEl resultado de restar la matriz \n\n"**);**

//Invocamos a la función mostrar matriz, e indicamos

//que la matriz a mostrar es matriz\_uno

mostrar\_matriz**(**matriz\_uno**,** m**,** n**);**

//Enviamos mensaje

printf**(**"\n\nCon la matriz \n\n"**);**

//Invocamos a la función mostrar matriz, e indicamos

//que la matriz a mostrar es matriz\_dos

mostrar\_matriz**(**matriz\_dos**,** m**,** n**);**

//Mostramos mensaje

printf**(**"\n\nes: "**);**

//iniciamos un ciclo de repetición para las filas

**for(**i **=** 0**;** i**<**m**;** i**++){**

//Iniciamos un segundo ciclo de repetición para columnas

**for(**j **=** 0**;** j**<**n**;** j**++){**

//Realizamos la resta entre matrices y almacenamos en

//nuestra matriz resultante

matriz\_resta**[**i**][**j**]** **=** matriz\_uno**[**i**][**j**]** **-** matriz\_dos**[**i**][**j**];**

**}**

**}**

//Mostramos la matriz resultante

mostrar\_matriz**(**matriz\_resta**,** m**,** n**);**

**}**

**}**